PROPUESTA METODOLÓGICA PARA LA CREACIÓN E IMPLEMENTACIÓN EN PRODUCCIÓN DE UN MODELO DE OTORGAMIENTO CREDITICIO UTILIZANDO VARIABLES ALTERNATIVAS EN UNA ENTIDAD FINANCIERA.

ANDRÉS FERNÁNDEZ PALACIO

Trabajo de grado para optar al título de

INGENIERO DE SISTEMAS Y COMPUTACIÓN

ALEJANDRO PEÑA PALACIO PH. D

UNIVERSIDAD EIA
INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
ENVIGADO
2018

Dedicado a mi familia y a Wilson Palacio

AGRADECIMIENTOS

A todos los profesores que hicieron parte de mi carrera, a mis compañeros, a Johan y a mi familia.

CONTENIDO

pág.

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ANEXOS

RESUMEN

El presente trabajo está enfocado en una Fintech de un banco importante colombiano el cual luego de la flexibilización de la ley colombiana para el otorgamiento de microcréditos decidió sacar al mercado un producto relacionado enfrentando dos retos principales: ¿Cómo calificar a las personas preguntándoles la menor cantidad de información posible (experiencia de usuario)? El segundo es: ¿Cómo consumir el modelo creado en un servicio de alto rendimiento que permita las consultas en línea y adaptándose a la arquitectura de software ya existente? En este trabajo se presenta una propuesta para la solución de ambos retos.

Para el primer reto enfocado al modelo se creó un modelo basado en el uso de los usuarios en la aplicación sin la necesidad de preguntarles ninguna información adicional aparte de la otorgada en el momento del registro, para la creación del modelo se utilizó un algoritmo de gradiente descendente conocido como Xgboost y su adaptación a SageMaker. Como resultado se obtuvo una precisión de 65% frente al 63.46% del modelo que había desarrollado Lenddo, un proveedor contratado por la Fintech.

Se llegó a la conclusión de que los modelos que utilizan variables alternativas para el otorgamiento no son una buena opción para ser usados como modelo principal de scoring sino como un complemento a los modelos tradicionales que sirva para aumentar la inclusión financiera. Se sugirió utilizar un modelo que se base en el incremento gradual de confianza y utilizar Lenddo únicamente como una fuente de captura de información alternativa para otros fines.

En lo referente al sistema, Amazon SageMaker fue el ganador de un proceso que utilizando ingeniería de requisitos estableció las necesidades de la Fintech y posteriormente evaluó las distintas posibles soluciones a través de un análisis de 8 variables, SageMaker obtuvo la calificación más alta de los evaluados. El sistema fue implementado exitosamente y no sólo podrá ser usado para el consumo masivo del modelo de crédito sino para cualquier otro modelo que quiera implementar la Fintech en un futuro.

ABSTRACT

The present work is focused on a Fintech of a major Colombian bank which after the flexibilization of the Colombian law for the granting of microcredits decided to bring to the market a related product facing two main challenges: How to qualify people by asking them the least amount of possible information (user experience)? The second is: How to consume the model created in a high-performance service that allows online queries and adapting well to the existing software architecture? In this work a proposal for the solution of both challenges is presented.

For the first challenge focused on the model, a model was created based on the use of the users in the application without the need to ask them for any additional information other than that granted at the time of registration, for the creation of the model a gradient descending algorithm was used known as Xgboost and its adaptation to SageMaker. As a result, an accuracy of 65% was obtained, compared to 63.46% of the model developed by Lenddo, a supplier hired by the Fintech.

It was concluded that models that use alternative variables for granting are not a good option to be used as the main scoring model but as a complement to traditional models that serve to increase financial inclusion. It was suggested to use a model based on the gradual increase of confidence and to use Lenddo only as a source of alternative information capture for other purposes.

Regarding the system, Amazon SageMaker was the winner of a process that using requirements engineering established the needs of the Fintech and later evaluated the different possible solutions through an analysis of 8 variables, SageMaker obtained the highest rating of those evaluated. The system was successfully implemented and can not only be used for the massive consumption of the credit model but for any other model that wants to implement the Fintech in the future.

INTRODUCCIÓN

Luego de que en el año 2014 el ministerio de hacienda de Colombia decidiera aprobar la modalidad de los microcréditos y diera libertad a las entidades financieras para utilizar cualquier método en la evaluación del otorgamiento se creó un nuevo mercado compuesto principalmente por las Fintech dispuestas a ofrecer este tipo de préstamos, esto trajo consigo distintas formas de evaluar un cliente para un crédito y retos en cuanto a la implementación de un modelo analítico en un ambiente de producción para un consumo masivo. El presente trabajo presenta una propuesta a una Fintech asociada a un banco importante del país la cual cuenta con los retos descritos.

La forma en como la Fintech consideraba resolver el problema de la calificación de clientes de los cuales tiene muy poca información financiera asociada es mediante el uso de variables alternativas a través de un tercero embebido en la aplicación para la creación del score de crédito, en este trabajo se evalúa la pertinencia del modelo otorgado y se hace un enfoque relacionado con la forma en que los usuarios utilizan la aplicación de la Fintech.

Para el segundo reto asociado a la forma en como se debe implementar y consumir estos modelos de crédito no existe una metodología o herramienta predominante en la literatura, la Fintech tampoco cuenta con ningún sistema para esto por lo cual se construyó desde cero la implementación de este sistema partiendo de la licitación de requisitos de las distintas partes interesadas hasta el despliegue final del modelo en la plataforma.

Los resultados del trabajo se dividen en dos partes: los del modelo y los de la implementación del sistema. En el modelo se destaca la creación de un modelo utilizando las variables de uso de los usuarios en la aplicación el cual obtuvo una precisión mayor que la otorgada por el proveedor (65% frente al 63.46%). Por el lado del sistema se destaca la implementación de SageMaker de forma exitosa quedando apto para la implementación de cualquier otro modelo en el futuro.

Por último, estos resultados obtenidos se lograron mediante el uso de un algoritmo de gradiente descendente conocido como Xgboost y su implementación en sagemaker en cuanto lo referente al modelo. En cuanto a la implementación del sistema, las necesidades de la Fintech fueron recogidas utilizando ingeniería de requisitos y detallando las arquitecturas existentes, para la elección del sistema se utilizó una calificación de 8 variables de las posibles soluciones donde Amazon SageMaker obtuvo la calificación más alta

1. PRELIMINARES

1.1 PLANTEAMIENTO DEL PROBLEMA

Los scoring crediticios han sido la herramienta que le ha permitido a las instituciones financieras determinar el nivel de riesgo de default que tiene un cliente. Según Crook, Edelman, & Thomas (2007) hay tres tipos de modelos de riesgo crédito, los que estiman la probabilidad de default(PD), los que miden la exposición al default(EAD) y los que miden la perdida esperada del default. El PD son los que han gozado de mayor interés históricamente debido a que han impulsado áreas del conocimiento como la estadística y el aprendizaje de máquina, así mismo es el tipo en el que se centrará este trabajo.

Luego de que en el año 2014 el ministerio de hacienda de Colombia decidiera aprobar la modalidad de los microcréditos y diera libertad a las entidades financiera para utilizar cualquier método en la evaluación del otorgamiento ("Decreto 2654 de 2014 Nivel Nacional," n.d.), se creó un nuevo mercado compuesto principalmente por las Fintech dispuestas a ofrecer este tipo de préstamos. Con la libertad otorgada para la evaluación crediticia por parte del estado surgen principalmente dos retos, el primero es: ¿Cómo calificar a las personas preguntándoles la menor cantidad de información posible (experiencia de usuario)? El segundo es: ¿Cómo consumir el modelo creado en un servicio de alto rendimiento que permita las consultas en línea y adaptándose a la arquitectura de software ya existente en una Fintech? (Y. Lee, Scolari, Interlandi Microsoft Redmond, Weimer, & Chun, n.d.)

Actualmente existe una gran cantidad de algoritmos de inteligencia artificial, entre algunos de los más usados en la predicción del score crediticio se encuentran: las redes neuronales (West, 2000), las maquinas vector soporte (Huang, Chen, & Wang, 2007; Min & Lee, 2005)y los arboles de decisión (T.-S. Lee, Chiu, Chou, & Lu, 2006; Nie, Rowe, Zhang, Tian, & Shi, 2011). Es importante destacar que a pesar del avance en machine learning durante los últimos años los métodos estadísticos tradicionales siguen siendo populares debido a su facilidad de implementación y precisión(Lessmann, Baesens, Seow, & Thomas, 2015)

Este trabajo pretende generar una propuesta para los retos mencionados, tomando como base para la metodología a desarrollar los datos de clientes y la arquitectura de una Fintech que se prepara para otorgar microcréditos en el mercado colombiano.

1.2 OBJETIVOS DEL PROYECTO

1.2.1 Objetivo General

Proponer un modelo de otorgamiento crediticio que utilice variables alternativas para la predicción y definir la forma en la cual este será implementado en un ambiente de producción en una entidad financiera.

1.2.2 Objetivos Específicos

- Analizar los parámetros y variables a utilizar en el modelo sobre la información de los usuarios y sobre la organización en su arquitectura existente.
- Establecer las necesidades de la organización a la hora de consumir e implementar el modelo.
- Definir un proceso para la implementación en producción de un modelo creado
- Validar la implementación y resultados del modelo.

1.3 MARCO DE REFERENCIA

1.3.1 Antecedentes

Los modelos de predicción de la probabilidad de default han sido calculados de diversas formas a lo largo del tiempo, hasta ahora no existe una metodología estándar para la realización e implementación de estos. Tradicionalmente se han utilizado las scorecards que en conjunto con los modelos logísticos y distintas técnicas estadísticas se han implementado para determinar la probabilidad de impago de un grupo de personas con características similares, estas estaban orientadas a un estudio en base a su historial crediticio(Universidad Nacional de Colombia. Departamento de Economía., Universidad Nacional de Colombia. Departamento de Teoría y Política Económica., & Universidad Nacional de Colombia. Escuela de Economía., 2013).

Los métodos estadísticos que han predominado han sido el LDA (Altman, 1968) y regresiones logit (Wiginton, 1980) ,ambos autores obtuvieron buenos resultados y durante mucho tiempo fueron un estándar en las entidades financieras. Con el avance de la computación el uso de técnicas relacionas con el aprendizaje de maquina empezaron a ganar terreno en esta área, es así como West (2000) utilizó redes neuronales para la predicción y posteriormente Huang et al., (2007) realizaron una aproximación utilizando maquinas vector soporte. Más recientemente Nie et al., (2011) utilizaron arboles de decisión para la predicción. Es importante resaltar que todas las aproximaciones mencionadas han tenido buenos resultados en el campo. En el año 2015 el ensamble de modelos para este tipo de análisis empezó a tomar relevancia a partir de un documento publicado que exponía un benchmarking con el estado del arte para algoritmos de clasificación referentes a credit scoring, dentro de las técnicas más populares y poderosas del momento se pueden encontrar el random forest y el xgboost (Lessmann et al., 2015).

En cuanto a la implementación de este tipo de modelos en un ambiente de consumo masivo, Kalechofsky(2016) reconoce la importancia de la integración del modelo con la arquitectura existente en una organización, además crea el concepto de la arquitectura cognitiva la cual identifica la tecnología, arquitectura e infraestructura, así como el soporte administrativo y financiero para soportar las prácticas de ciencia de datos, analítica predictiva y procesos de modelado. El autor define además cinco estrategias para el manejo de datos en una organización y se definen las tres capas de una arquitectura básica orientada a la construcción de modelos predictivos y analítica.

Y. Lee et al. Presentaron una propuesta para la utilización de sistemas de servicios predictivos de alto rendimiento basando en gráficos acíclicos dirigidos, en el proceso identifican 3 requisitos esenciales de rendimiento para que un modelo de aprendizaje de maquina pueda ser consumido por los usuarios de forma rentable. Se destaca también el uso de herramientas como Clipper un servidor de baja latencia enfocado en la facilidad de despliegue (Crankshaw et al., n.d.) y Rorodata que funciona como una api y está orientada al desarrollo, despliegue y administración de los modelos analíticos como servicios(PaaS)(Chitipothu Rorodata, n.d.)

Por último, en cuanto a los antecedentes por uso de variables alternativas para el credit scoring, se destacan dos: Cignifi y Lenddo EFL. Cignifi surge de una investigación en África la cual estableció una relación entre la forma de uso de los dispositivos móviles por parte de las personas y su disposición a ahorrar en los bancos tradicionales(“Mobile Phone Data as the Key to Promoting Financial Inclusion,” n.d.). Por otro lado, Lenddo surge con la idea de generar inclusión financiera, de darle más oportunidades a las personas que su historial crediticio no los favorece o simplemente no lo tienen, también ofrecen servicios de verificación de identidad. (Lenoir, n.d.)

1.3.2 Marco teórico

Para entender este trabajo es importante empezar por aclarar que son los modelos de riesgo crédito. Según Crook, Edelman, & Thomas (2007) hay tres tipos de modelos de riesgo crédito, los que estiman la probabilidad de default(PD), los que miden la exposición al default(EAD) y los que miden la perdida esperada del default o impago. Por este motivo el puntaje crediticio es una herramienta fundamental para las entidades financieras a la hora de otorgar un crédito ya que les permite maximizar el monto disminuyendo las perdidas, abarcando la mayor cantidad de personas posible.

Dentro de la metodología del estudio crediticio es necesario maximizar los verdaderos positivos a este porcentaje se le conoce como TRP o sensibilidad y los verdaderos negativos cuyo porcentaje es conocido como TNR o especificidad.(Singh & Rani, 2011)

Durante el proceso de creación del modelo existen varios algoritmos de aprendizaje de máquina que podrían ser usados, entre los tres más representativos están:

- Redes Neuronales: La motivación detrás de estas surge de los complejos sistemas neuronales detrás del aprendizaje biológico. Su uso es muy amplio, por ejemplo, para la detección de fraudes se suelen utilizar tres capas (entrada, oculta y salida). La entrada de una red neuronal suele ser un vector de características y la salida es la probabilidad para cada elemento de ese vector de ser un fraude o de caer en default en este caso.(Bolton, 2010) Aunque el uso de las tradicionales tres capas ha demostrado buenos resultados, las redes neuronales tienen algunas desventajas, por ejemplo, se debe seleccionar y ajustar la estructura de la red, el número de estados ocultos debe ser especificado y optimizado, además el rendimiento del clasificador es muy sensible al vector de entrada por lo que puede requerir de un preprocesamiento previo de las variables.(Mitchell, n.d.)

- Máquinas de vector soporte: Implementan el principio de la estructura de riesgo minimizado mediante la construcción del hiperplano óptimo. Entre sus ventajas se encuentra que no necesitan datasets muy grandes y el entrenamiento converge a una nica solución global, por este motivo suelen ser utilizados para la detección de fraudes en tarjetas de crédito. Entre sus desventajas están que son más costosos de implementar, no son de fácil interpretación y se debe especificar una gran cantidad de parámetros al algoritmo.

- Arboles de decisión: Es un método para aproximarse al valor de una función objetivo, donde la función de aprendizaje es representada por un árbol de decisión. El proceso de clasificación ordena un árbol desde la raíz con el fin de generar reglas de relacionamiento entendibles entre los atributos de entrada y la variable objetivo. La variable con mayor poder de predicción se encuentra en el primer nodo del árbol. Entre sus ventajas se encentra la facilidad de interpretación en contraposición es muy sensible al sobrentrenamiento (overfitting) (Mitchell, n.d.)

Como mencionó Lessman (2015) los algoritmos de boosting han tenido un fuerte crecimiento en los últimos años, por eso se describe a continuación uno de los algoritmos de mayor crecimiento, Xgboost.

Xgboost debe su nombre a eXtreme Gradient Boosting, surgió como una implementación de los algoritmos de boosting realizada por Tianqi Chen en C++ y posteriormente con el apoyo de múltiples colaboradores se transformó en una librería para distintos lenguajes como R, Python o Scala.

Entre sus características del sistema está:

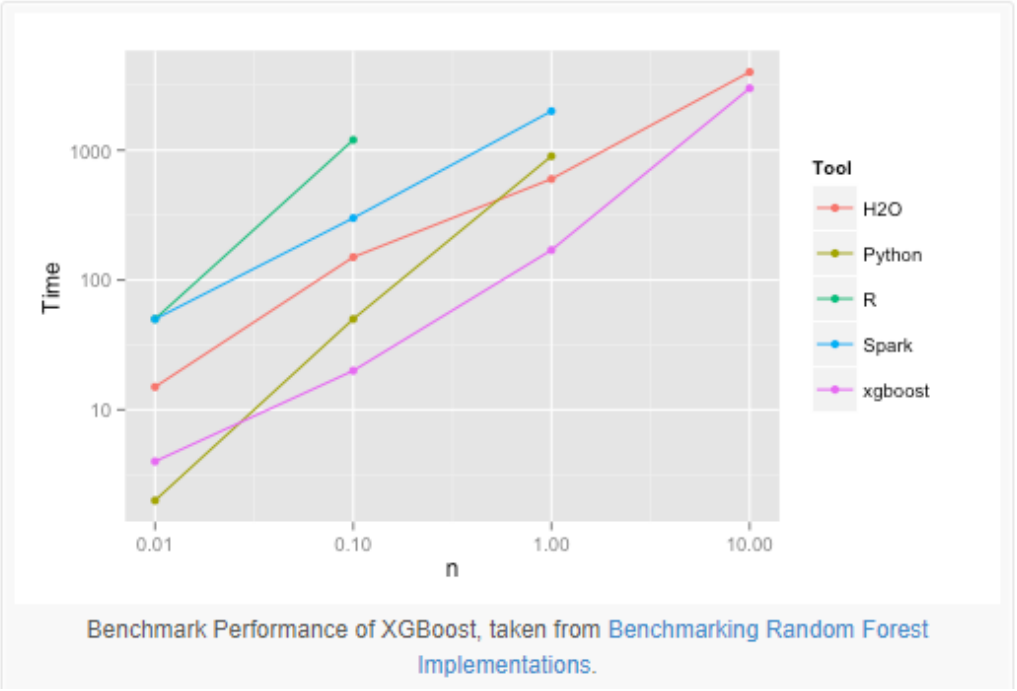
- Paralelización
- Computación distribuida
- Computación out-of-core (para conjuntos de datos muy grandes que no caben en memoria)
- Optimización de caché

Entre las características algorítmicas está:

- Manejo automático de datos nulos
- Estructura en bloque para soportar la paralelización
- Entrenamiento continuo, mejora un modelo ya entrenado con nuevos datos

Las principales razones que se destacan a la hora de utilizar Xgboost es su velocidad de ejecución y rendimiento del modelo.

En cuanto a su velocidad Xgboost ha demostrado ser mucho más rápido que otras implementaciones en otros lenguajes como se ve a continuación:



Benchmark Performance of XGBoost, taken from Benchmarking Random Forest Implementations.

Figura 1: Como se puede observar, Xgboost obtuvo un rendimiento mejor en casi todas las mediciones entregadas.

En cuanto a su rendimiento, el algoritmo ha sido recientemente el dominador de las distintas competiciones de Kaggle, una web dedicada al concurso de problemas de machine learning en todo el mundo. (Brownlee, 2016)

Respecto a la metodología e implementación de los modelos, existen cinco áreas claves para manejo exitoso de los datos en una organización:

1. Calidad de los datos: Se refiere a mantener datos certeros, convenientes y limpios.

2. Integración de datos: Como integrar las distintas fuentes de datos de forma óptima.

3. Gobierno de los datos: Creación de políticas para el almacenamiento y uso de los datos.

4. Federación de los datos: como proveer un acceso seguro y federado a los datos.

5. Manejo maestro de los datos: Como manejar eficientemente los datos

Cuando se trata de modelos predictivos se debe considerar una arquitectura de datos que pueda ser optimizada durante el tiempo. Deben de existir herramientas que permitan usar la salida de un modelo de manera eficaz en una aplicación tecnológica, desafortunadamente muy pocas empresas en el mundo tienen una arquitectura óptima para la analítica ya que no sólo se requiere una gran capacidad de almacenamiento sino

también velocidad. La arquitectura debe ser capaz de proveer la capacidad necesaria a la analítica sin sacrificar los demás procesos tecnológicos. (Kalechofsky, 2016)

Los elementos claves durante el desarrollo de un modelo predictivo en un ambiente de producción son:

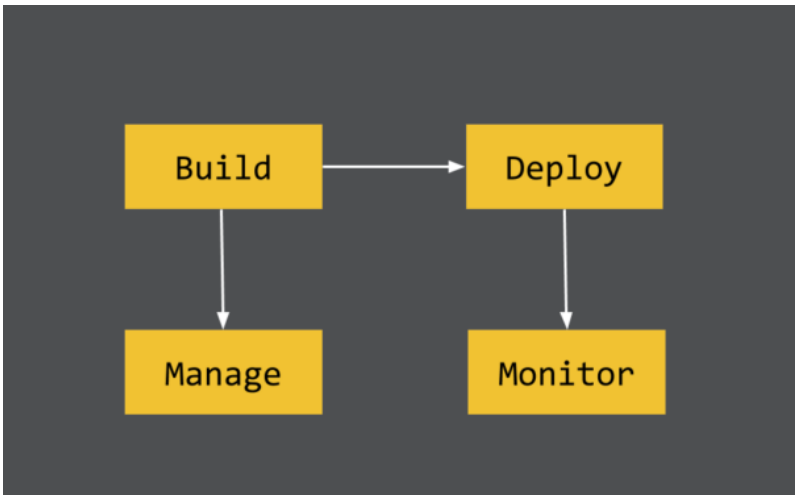


Figura 2: Se observan los elementos claves a tener en cuenta durante el uso de un modelo analítico en un ambiente de producción Imagen de (Chitipothu Rorodata, n.d.)

Durante la construcción del modelo se debe tener en cuenta la instalación de las dependencias necesarias y la automatización del reentrenamiento periódico. En la fase del despliegue es necesario documentar claramente la forma de consumo la configuración de los endpoints y lo más importante de todo: el modelo se debe poder escalar de acuerdo con los cambios en el consumo. En cuanto a la administración se debe llevar un seguimiento de las distintas versiones del modelo y un marco para el trabajo de colaboración en equipo. Por último, en cuanto a monitoreo se debe supervisar la precisión en producción y el rendimiento frente al tiempo de respuesta de una petición.(Chitipothu Rorodata, n.d.)

2. METODOLOGÍA

2.1 CASO DE ESTUDIO:

La metodología con la que se realizó este trabajo se divide en cuatro fases principales. La primera fase de la metodología se divide en dos aspectos: el primero corresponde a una parte técnica encargada de seleccionar y analizar las variables a utilizar en el modelo crediticio, así como su recolección.

Para la recolección de los datos se empezará con establecer y describir el estado actual de Lenddo, el cual es un proveedor de la Fintech encargado de la recolección de datos alternativos en la aplicación y estudiar su propuesta respecto a créditos.

Lenddo se encuentra embebido en la aplicación y se encarga de recolectar información de los dispositivos en donde se encuentra instalado, esta información consiste en:

- Cantidad de SMS entrantes y salientes
- Largo de SMS entrantes salientes
- Información de la agenda
- Acceso a la lista de contactos
- Acceso al registro de llamadas
- Información general del dispositivo (Porcentaje de batería, tipos de aplicaciones instaladas, etc.)

Es importante resaltar que este tercero sólo funciona en dispositivos Android y donde el usuario haya dado su autorización.

Lenddo tiene principalmente tres misiones:

- Recolectar datos sobre el comportamiento de los usuarios
- Ofrecer un servicio que permita la validación de personas para prevenir el fraude por suplantación
- Otorgar un score propio asociado a la probabilidad de impago de los clientes

De sus tres misiones, para este trabajo se resalta la asociada a la creación del Score propio.

El segundo aspecto es un análisis descriptivo cuyo objetivo es entender como es la arquitectura existente en la organización. Para ello se expondrán los distintos servicios que utiliza la aplicación posteriormente se describirán dos arquitecturas fundamentales, la primera es la arquitectura principal de toda la aplicación que se compone de tres capaz, la segunda describirá como funciona los distintos flujos de datos disponibles dentro de la aplicación.

La segunda fase consiste en la manera mediante la cual se establece cuáles son las necesidades de la Fintech respeto al sistema encargado de desplegar en producción los sistemas analíticos, para este caso el modelo de créditos, del cual también se establece que se espera de él. Para lograr este propósito se utilizarán dos herramientas fundamentales de la ingeniería de requisitos: los requisitos funcionales y no funcionales. Es importante resaltar que el mayor aporte de este trabajo se encuentra en la solución a los requisitos no funcionales.

Para este caso los requisitos funcionales se refieren a cuál es el resultado o comportamiento esperado del modelo de créditos, los requisitos no funcionales son los que ayudan a delimitar el problema al contexto de la Fintech mediante sus limitaciones o necesidades.

La tercera fase consiste en exponer como se realiza el proceso de selección de la solución tentativa para la Fintech y una vez se cuente con un candidato, se expone el proceso a seguir para poner en producción el modelo de crédito.

Para el proceso de implementación del sistema se describirán las distintas fases que se requieren para desplegar el modelo teniendo en cuenta la arquitectura existente. Por último, se establece como se valida que el sistema y el modelo cumplan con lo esperado por la Fintech. Con el fin de validar la implementación y resultados del modelo se procederá a evaluar el cumplimiento de los requisitos funcionales y no funcionales especificados.

2.2 METRICAS O RESULTADOS ESPERADOS

Para la primera fase correspondiente a la recolección de la información y el análisis descriptivo se deberá entregar un diagrama donde exponga la arquitectura de la aplicación de la Fintech, así como su arquitectura para datos, además se deberá realizar una descripción de los distintos servicios con los que cuenta la aplicación. Para Lenddo se realizará la investigación correspondiente y se entregará el score obtenido por ellos. Además, se debe establecer cuál es la postura de la Fintech respecto a su uso posterior en el modelo analítico.

Para el descriptivo estadístico se hará uso de diagramas de caja y bigotes, el objetivo de este paso es establecer que variables con las que cuenta la Fintech podrían o no ser usadas en la construcción del modelo.

La segunda fase correspondiente al establecimiento de las necesidades entregará el listado de requisitos funcionales (correspondientes al modelo) y no funcionales (correspondientes al sistema e implementación).

Para la selección del proceso, correspondiente a la tercera fase, cada una de las siguientes variables se calificó en la escala de uno a tres donde uno es la calificación más baja y tres la más alta:

- Disponibilidad
- Escalabilidad
- Facilidad de implementación
- Despliegue
- Compatibilidad
- Mantenimiento y monitoreo
- Concurrencia
- Modelos

Disponibilidad: Es la capacidad del sistema de responder a una petición en cualquier momento teniendo en cuenta el respaldo en caso de fallas.

Escalabilidad: Capacidad del sistema de aumentar o disminuir su potencia con el fin de satisfacer distintos niveles de demanda a un coste óptimo.

Facilidad de implementación: Se refiere a la cantidad de conocimientos nuevos o herramientas que el equipo de analítica debe dominar con el fin de realizar una implementación adecuada.

Despliegue: Es la capacidad del sistema de realizar un despliegue del modelo, la facilidad del consumo de parte de la aplicación es tenida en cuenta en esta variable.

Compatibilidad: Se refiere a la facilidad con que el sistema se acopla a la arquitectura existente de la Fintech.

Mantenimiento y monitoreo: Se refiere a la facilidad con que se puede monitorear el sistema y mantener los modelos alojados en él, la generación de alertas se tiene en cuenta en esta variable.

Concurrencia. Capacidad del sistema para manejar grandes cantidades de peticiones al día.

Modelos: Capacidad del sistema para entrenar y/o alojar modelos de distintos tipos.

Teniendo en cuenta lo anterior se genera la siguiente matriz para el análisis de las distintas ofertas disponibles:

Servicio	Disponibilidad	Escalabilidad	Implementacion	Despliegue	Compatibilidad	mant y monit	Concurrencia	Modelos	Total
Clipper									
Rorodata									
Plumber									
Flask									
Sagemaker									

Figura 3: Matriz de puntuación de las opciones disponibles

Por último, para realizar la evaluación se procederá a calificar el cumplimiento de cada requisito en una escala de uno a cinco, donde uno se refiere a un cumplimiento nulo y cinco a un cumplimiento en su totalidad. A su vez para cada uno de los requisitos tanto funcionales como no funcionales se deberá justificar dicha calificación. Para los resultados del modelo se presentará una puntuación AUC y una matriz de confusión.

En el siguiente diagrama se observan las distintas fases de la metodología y sus componentes principales:

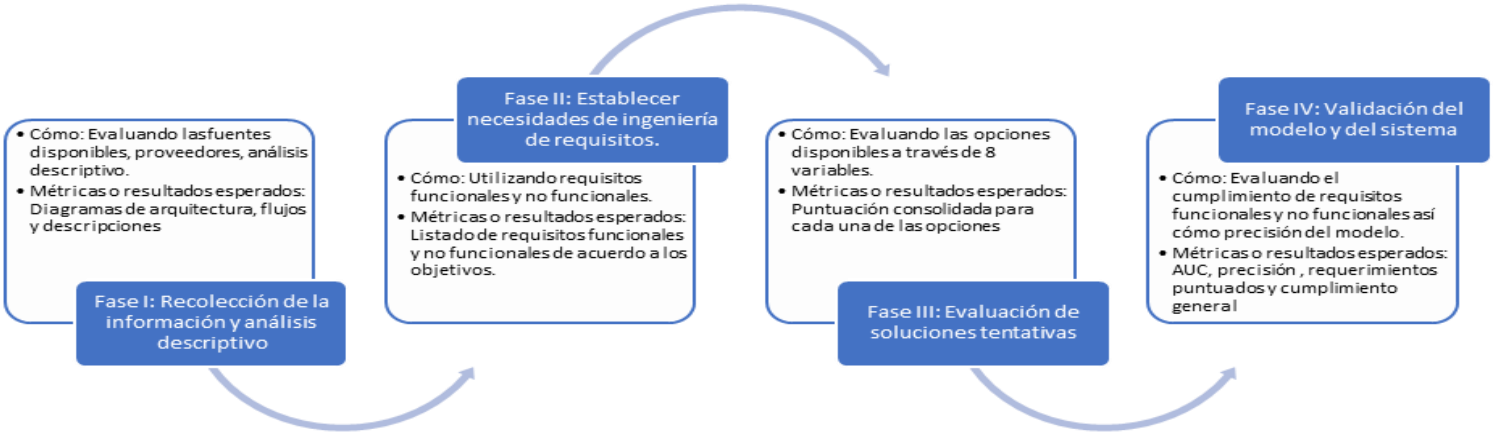


Figura 4: Las cuatro fases de la metodología empleada.

3. PRESENTACIÓN Y DISCUSIÓN DE RESULTADOS

3.1 RECOLECCION DE LA INFORMACIÓN

3.1.1 Score de Lenddo

Para la creación de su score Lenddo dividió a los usuarios en seis grupos de riesgo o bandas. La banda cero corresponde al grupo con menor probabilidad de default, mientras que la 5 a la de mayor probabilidad.

Para hacer esto se utilizó sólo la información de aquellos que contaban con una calificación en las centrales de riesgo (función objetivo), tuvieran Android y además hayan autorizado la captura de su información. Es importante tener en cuenta que la cantidad de usuarios de la aplicación es de aproximadamente 500000 mientras que la cantidad de usuarios que cumplieron con los filtros realizados por Lenddo fue apenas un poco más de 1200.

La primera corrida del modelo se realizó a finales del 2017, para ello se utilizaron diversos algoritmos tales como un árbol, una SVM o modelos logit. Durante esta corrida se realizaron test OOS que dio como resultado una precisión a una banda del 20.3%.

Lenddo siguió capturando datos y en mayo de 2018 con casi 1500 registros volvió a correr el modelo, esta vez no se utilizaron varios algoritmos para la predicción sino uno solo de tipo boosting. Se realizaron pruebas OOS y OOT (con los datos de finales de 2017) de esta forma se validaba que el modelo no se estuviera sobreentrenando y que el contexto no haya cambiado lo que supondría un replanteamiento de todo el proceso.

Con los datos de mayo y aplicando la nueva técnica se obtuvo el siguiente resultado OOT:

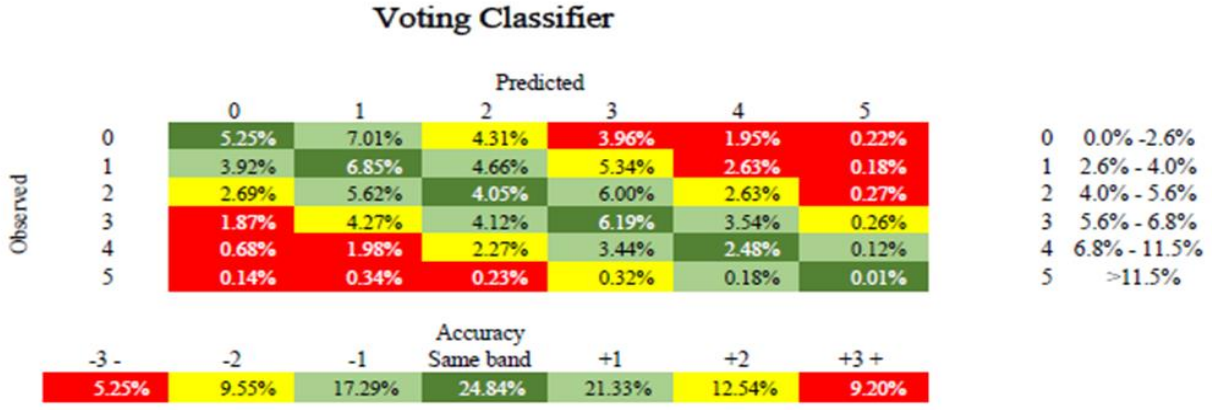


Figura 5: Resultados del score de Lenddo.

Como se observa Lenddo obtuvo una precisión a una banda de casi el 25%, si tenemos en cuenta que son seis grupos entonces tener esta precisión de forma aleatoria corresponde a 1/6 o 16.67%.

Al analizar la importancia de las variables se obtuvo que lo que más pesó fue la cantidad y largo de los SMS, posteriormente se concluyó que el modelo terminó prediciendo quien era prepago o pospago y debido a la relación implícita de esto con el comportamiento crediticio de una persona se obtuvo los resultados expuestos.

Debido a que el score de Lenddo no fue convincente, el negocio se propuso a crear un score propio con la información disponible de uso de la aplicación por parte de los usuarios.

3.2 ANALISIS DESCRIPTIVO DE LAS VARIABLES

3.2.1 Fuentes de datos

La Fintech donde se realizó este trabajo forma parte de un banco tradicional importante en el país. Por este motivo se tomaron los clientes comunes con el banco que tuvieran una calificación crediticia. La clasificación en grupos de riesgo que utiliza el banco es la siguiente:

Escala maestra	G1	G2	G3	G4	G5	G6	G7	G8
PD inicial	0.0%	0.9%	1.7%	3.0%	5.0%	8.0%	10.0%	28.0%
PD Final	0.9%	1.7%	3.0%	5.0%	8.0%	10.0%	28.0%	100.0%

Figura 6: Clasificación de los grupos de riesgo otorgados por el banco.

Debido al carácter exploratorio del análisis descriptivo se agrupó a los G1 y G2 como 1 (buenos) y a los demás como 0 (malos). Esto permite verificar de forma más clara si una variable dada es discriminante o no. Además, permite enfocar la exploración no sólo en la construcción sino también en la implementación del modelo.

Luego de obtener esta información se procedió a aplicar distintos filtros con el fin de realizar el análisis descriptivo con una menor cantidad de ruido.

A continuación, se presenta el flujo de filtros realizados:

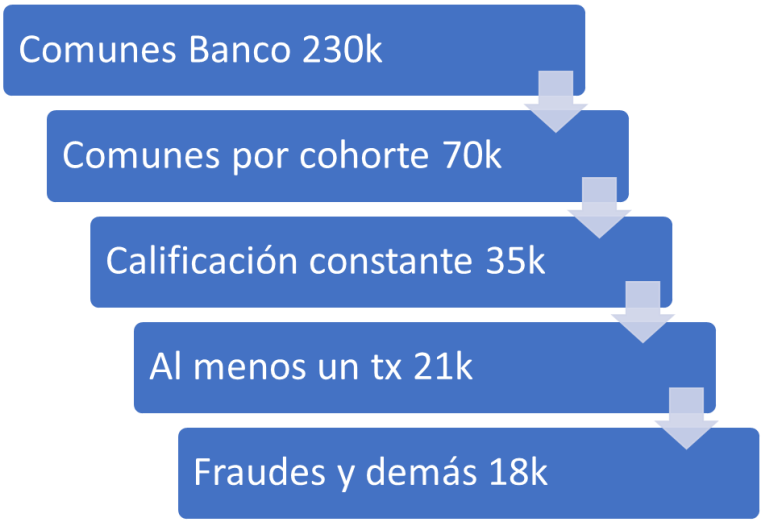


Figura 7: Filtros realizados de los datos entregados por el banco.

- Comunes Banco corresponde a la cantidad de usuarios comunes entre la Fintech y el banco a junio de 2018

- Comunes por cohorte se refiere a los comunes por mes sin duplicados, los 70 mil corresponden a los del mes de junio.

- Calificación constante son del grupo de 70 mil, aquellos cuya calificación fue igual en los 3 meses analizados por el banco. Esto con el fin de establecer un perfil fijo por persona al compararlo con el histórico que se tiene del usuario en la Fintech.

- Del grupo anterior se eliminó aquellos que no hubieran hecho ninguna transacción (uso o gestión del dinero), ya que la base del análisis se basa en el uso de la aplicación.

- Por último, se excluyó del análisis a aquellos reportados por fraudes, tuvieran menos de 1000 pesos en sus cuentas o dónde la captura de la información fue errada.

En base a lo anterior se tomaron las siguientes variables para el análisis:

- Valor Metas
- Valor Bolsillos
- Valor Guardadito
- Valor saldo cuenta
- Valor transacción promedio
- Número de transacciones
- Cashin
- Cashout
- Transferencia
- Marketplace
- SO
- Edad

Valor metas: Son ahorros con un objetivo (un carro, una moto, un juego, etc.) e incluso débitos automáticos dentro de la aplicación, está dado en pesos.

Valor bolsillos: Es la forma en cómo se organiza el dinero destinado para pagos o gastos comunes en la aplicación (arriendo, mercado, etc.). Está dado en pesos.

Valor guardadito: Son ahorros sin un objetivo específico, como un fondo de emergencia. Está dado en pesos.

Valor saldo cuenta: Valor disponible en la cuenta, agrupa el saldo libre y el saldo asignado en gestión del dinero (metas, bolsillos y guardadito). Está dado en pesos

Valor transacción promedio: Valor promedio de las transacciones realizadas por el usuario.

Número de transacciones: Conteo de transacciones acumuladas desde la apertura de la cuenta hasta la fecha del análisis.

Cash in: Es la agrupación de distintos conceptos de transacciones que ingresan dinero al sistema de la Fintech, ejemplo: Recargas desde ACH, Recargas por PSE, etc. Es un conteo

Cash out: Es la agrupación de distintos conceptos de transacciones que retiran dinero del sistema de la Fintech, ejemplo: Retiros en cajero, Envío a otros bancos, etc. Es un conteo

Transferencia: Agrupa los códigos de transacciones correspondientes a los envíos entre los usuarios de la aplicación (el dinero permanece en el sistema). Es un conteo

Marketplace: Es el conteo de transacciones realizadas en una sección de la aplicación con el mismo nombre o “armario”. Ejemplo: recarga de celular, DIRECTV o Xbox live, etc.

SO: Es el sistema operativo del usuario, Android o IPhone.

Edad: Edad del usuario.

Estas variables fueron obtenidas de registro transaccional de los usuarios en la aplicación, y del registro de cuentas de usuarios.

3.2.2 Descriptivo estadístico

Con el fin de determinar si existía o no diferencia entre los “buena paga” y los “mala paga” se realizó una comparación entre ambos grupos para cada una de las variables.

Al analizar el set de datos disponible, se verificó que la cantidad de datos nulos era muy grande para valor metas, valor bolsillos, valor guardadito por tal motivo se decidió agrupar estas tres variables en una sola conocida como valor gestión. A pesar de lo anterior en general todas las variables del modelo se veían afectadas por la inactividad de los usuarios. A continuación, se muestran tales comparaciones:

- Valor Gestión:

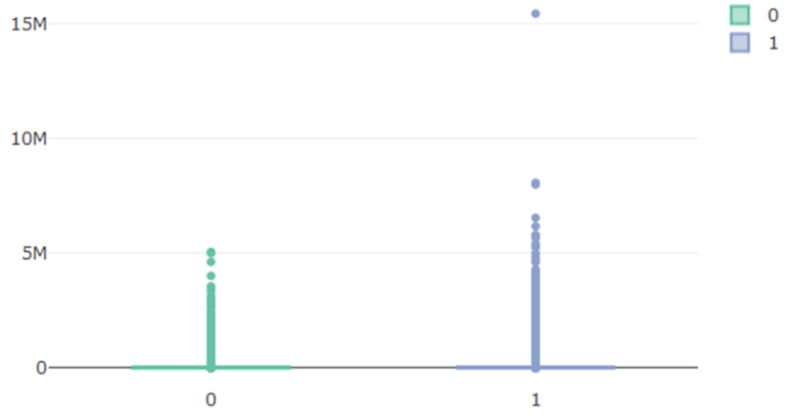


Figura 8: Media, mediana, moda y demás medidas estadísticas estaban todas en cero.

- Valor saldo cuenta:

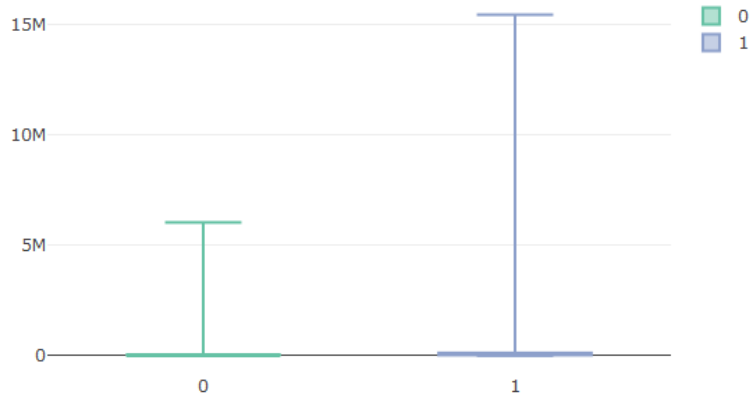


Figura 9: Un 50% de las buenas pagas tienen entre 7 y 100k en su cuenta mientras que los malas pagas tienen entre 2 y 12 mil pesos.

- Valor transacción promedio

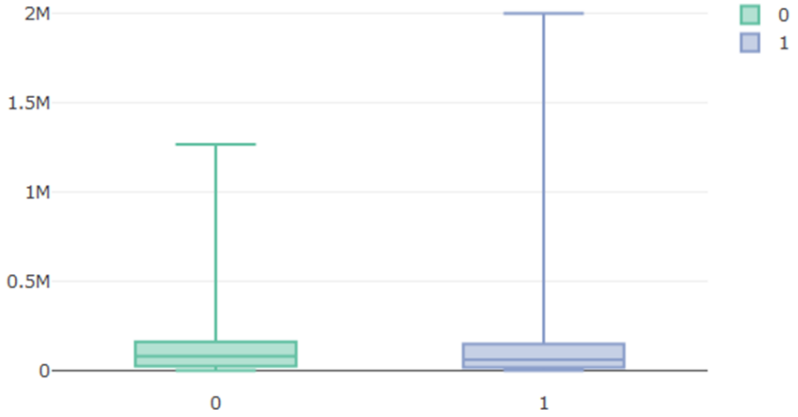


Figura 10: La mayoría de los buenos está entre 16 mil y 150 mil pesos. Los malos están entre 23 mil y 160 mil pesos. Como no existe una diferencia estadística se descarta la variable.

- Número de transacciones

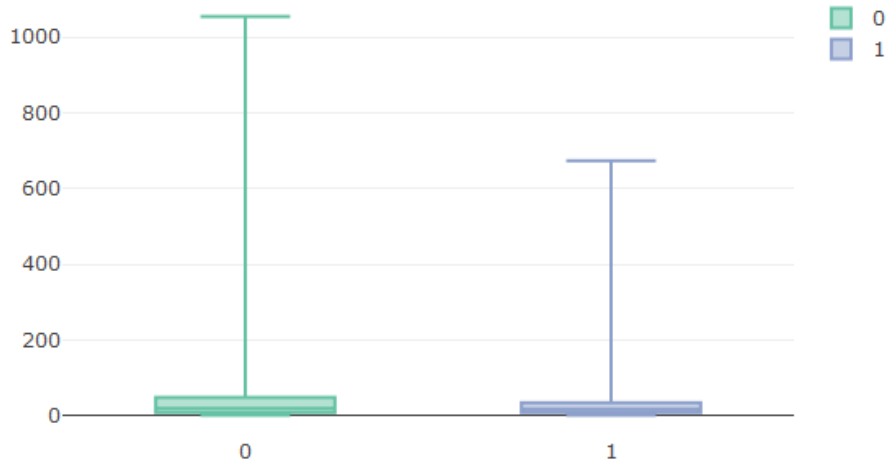


Figura 11: Los malas paga se encuentran ente 1 y 48 mientras los buenos entre 1 y 34.

- Cashin

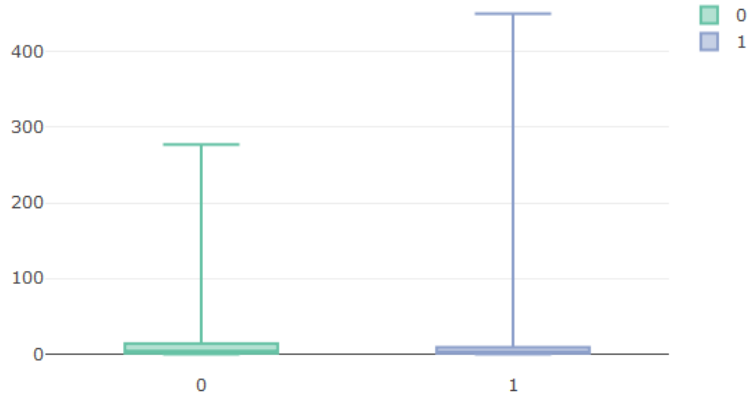


Figura 12: Los malas pagas realizan entre 0 y 14 mientras los buenas pagas entre 0 y 9..

- Cashout

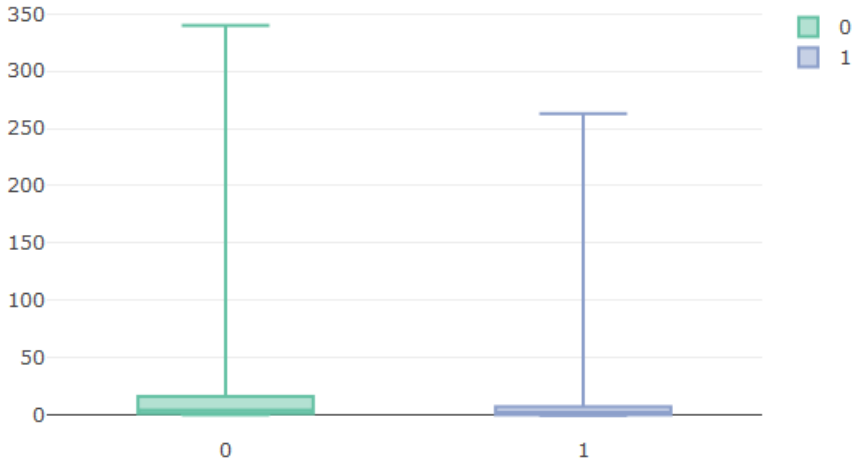


Figura 13: La mayoría de los malos están entre 1 y 16 y los buenos entre 1 y 7.

- Transferencia

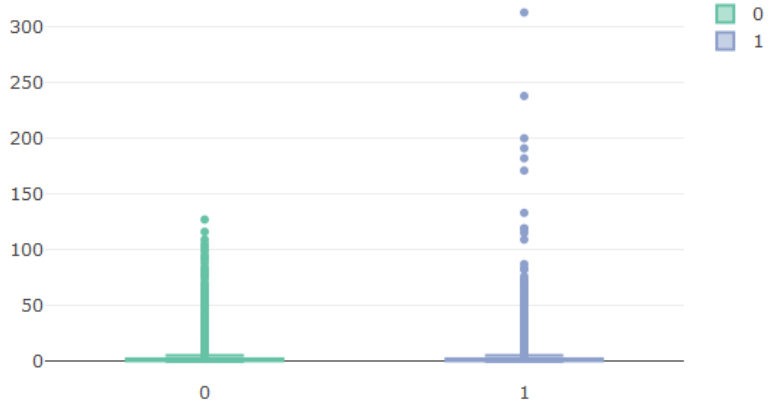


Figura 14: No hay evidencia de una diferencia entre ambos grupos. Para ambos casos la mayoría se encuentran entre 2 y 5 transferencias. Debido a que es una variable con pocos nulos, se le dio el "beneficio de la duda" y se agregó al modelo.

- Marketplace

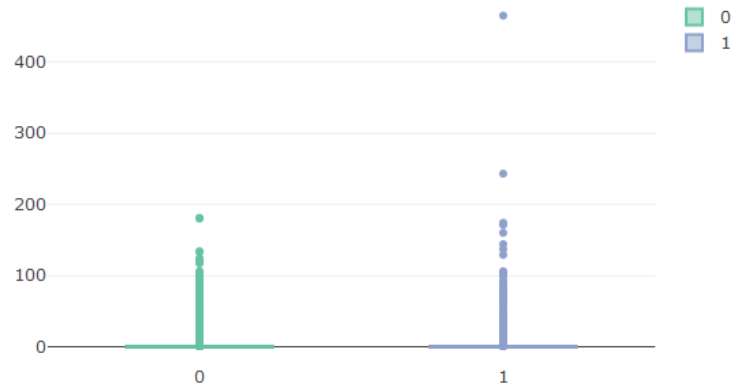


Figura 15: Es cero para ambos grupos, no se toma esta variable para el modelo.

- SO

  No existía ninguna diferencia significativa en cuanto a las proporciones de un sistema operativo en un grupo respecto al otro. Además de los 18.000 registros no teníamos esta información para 4.000 por lo cual se descartó.

- Edad

Figura 16: No se aprecia ninguna diferencia, la variable se descarta.

3.3 DESCRIPCIÓN DE LA ARQUITECTURA EXISTENTE

La arquitectura tecnológica de la Fintech tiene dos componentes principales, el primero es el Core bancario, de allí se obtiene la información transaccional y de cuentas de los usuarios, sin embargo, todo este sistema es muy complejo, su infraestructura y detalles son manejados por un tercero, además no es relevante conocerla para este trabajo.

El segundo componente se refiere a la arquitectura sobre la cual funciona la aplicación y más específicamente los distintos flujos de datos y procesos que son utilizados para analítica. La arquitectura principal es heredada de IBM y consta de tres capas: worklight, middleware o was y broker o bus. Dentro de cada componente funcionan distintos servicios en la nube y de Amazon en su mayoría, estos son:

- S3
- Cloudwatch
- Redshift
- DynamoDB
- Athena
- Lambda
- Glue
- Rekognition
- EC2
- VPC
- API Gateway
- EMR

Además de los anteriores que son exclusivos de Amazon es importante resaltar los siguientes open source:

- ElasticSearch

- Logstach
- Kibana
- Grafana
- Metabase
- Redash
- Spoon

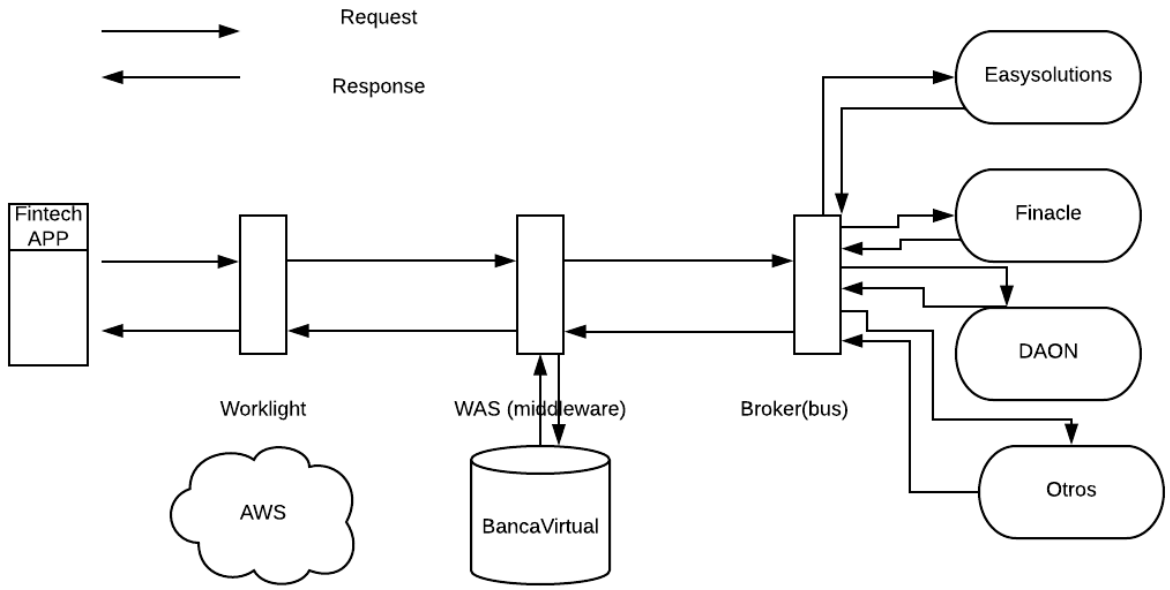


Figura 17: Diagrama de la arquitectura principal de la aplicación:

Cómo se puede observar es una arquitectura de tres capas, worklight se encarga de manejar todas las peticiones que hace el usuario desde el front. WAS llama al servicio necesario para procesar la petición por último bróker es el encargado de realizar y mantener las distintas conexiones de la aplicación con terceros.

En bancaVirtual se almacenan datos de los usuarios en la aplicación y la nube de AWS se refiere a los distintos servicios que son utilizados, esta se encuentra entre Worklight y WAS ya que estás son las dos capas que hace uso de esta.

Existen tres fuentes de datos fundamentales en el sistema de la Fintech. El primero es Finacle o el Core bancario, de este se obtiene el transaccional y el estado de cuentas de

los usuarios, se puede consultar en tiempo real a través de los logs de broker o todos los días en batch se recibe de parte de ellos el consolidado de transacciones diarias (mes a mes) y el consolidado de cuentas diarias.

El segundo es bancavirtual el cual es una base de datos en Oracle que funciona como la base de datos principal de la aplicación, como se mencionó anteriormente, allí se guarda todo lo referente al usuario en el momento de registro.

El tercero es un sistema mucho más complejo y son los flujos de datos que se obtienen a través de los logs de la aplicación en sus distintas capas o servicios, a continuación, se describe este proceso más detalladamente:

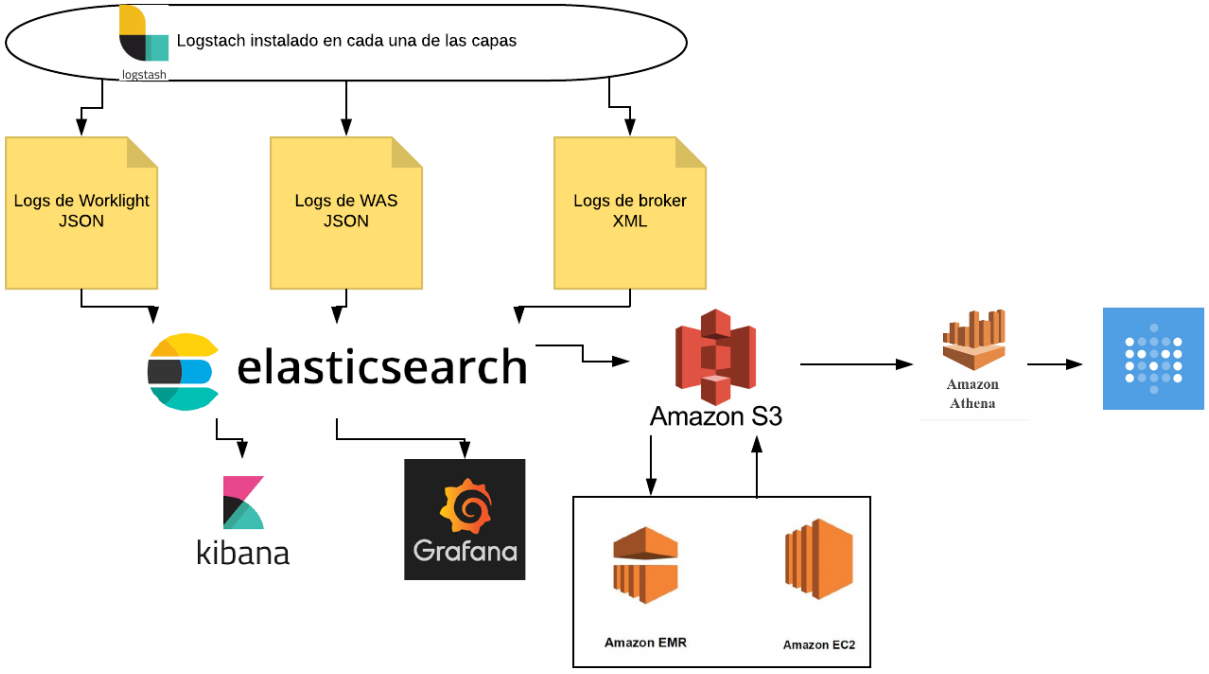


Figura 18: Diagrama de la arquitectura de datos de la aplicación:

El primer paso consiste en la captura de logs mediante beats y logstach instalado en cada una de las capas del sistema, estos capturan los logs y los envían a elasticsearch el cual es un motor de indexación y almacenamiento de los logs. Posteriormente hay dos caminos: El primero consiste en utilizar herramientas como kibana que permite visualización en texto o grafana que se especializa en monitoreo y gráficos con el fin de visualizar los logs en tiempo real como información no estructurada.

El segundo camino consiste en enviar los logs a S3 el cual es un sistema de almacenamiento de gran capacidad. Una vez allí se pueden correr procesos ETL que se encuentran en dos partes: Elastic Map Reduce que utiliza SCALA y SPARK para realizar el proceso en paralelo en un sistema distribuido o EC2 el cual no paraleliza el proceso,

sino que ejecuta los scripts en un computador de gran capacidad dónde todo se encuentra basado en Python o Spoon.

Sin importar el sistema de procesamiento utilizado la salida de ambos son archivos transformados a un bucket en s3, estos archivos se encuentran en formato CSV con el fin que desde Athena se puedan generar External Tables que los utilicen como insumo una vez definido el schema de la tabla. Por último, un programa open source llamado Metabase (el icono azul en el diagrama) se encarga de presentar la información de manera amigable y cualquier usuario (inteligencia de negocios).

3.4 CAPTURA DE REQUISITOS

Una vez descrito y entendido los componentes principales de la aplicación se procede a establecer las necesidades de la organización en cuanto al consumo de un modelo analítico, esto servirá para desarrollar una estructura que permita el manejo de cualquier tipo de modelo y no sólo para créditos como se ejemplifica en este trabajo.

La ingeniería de requisitos es posiblemente el componente más difícil en cualquier proyecto de software. Con el fin de mejorar este proceso se ha escrito mucho y por este motivo se adoptan dos conceptos claves de este dominio para describir las necesidades de la Fintech: los requisitos funcionales y no funcionales

Antes de exponer los distintos requisitos es importante tener en cuenta las partes interesadas en la solución como se muestra a continuación:

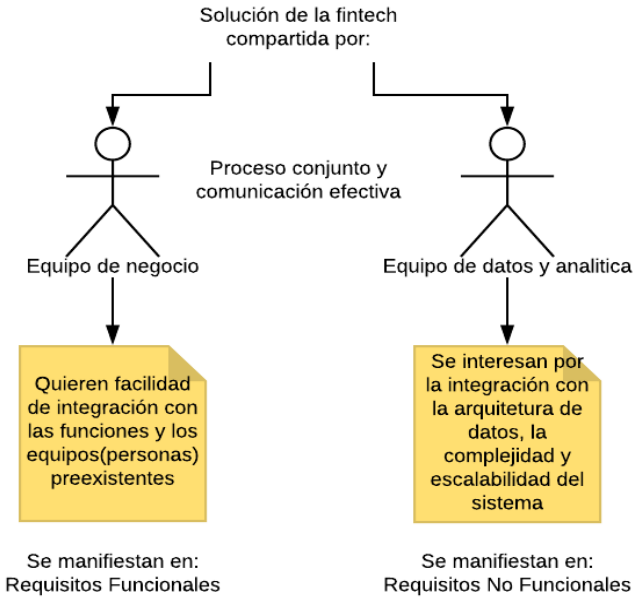


Figura 19: Diagrama de la arquitectura de datos de la aplicación

3.4.1 Requisitos funcionales

- Desde la aplicación el sistema deberá permitir a cada usuario conocer si su crédito fue aprobado o no.

- El sistema entregará información acerca de la calificación otorgada a una persona a personas de negocio con el fin de evaluar casos específicos.

- Si una persona que no ha solicitado un crédito es candidato a ser buena paga, el sistema deberá estar en la capacidad de identificarlo y entregar este insumo para generar recomendaciones en la aplicación.

3.4.2 Requisitos no funcionales

- El sistema deberá tener alta disponibilidad ya que una persona puede solicitar un crédito en cualquier momento.

- El sistema deberá ser fácilmente escalable en cuanto a su capacidad de procesamiento.

- El tiempo de respuesta no deberá ser mayor a 5 minutos.

- El sistema deberá ser capaz de manejar múltiples peticiones por día.

- El sistema deberá ser fácilmente adaptable a la arquitectura ya existente de la Fintech y el componente de API Gateway

- El sistema debe requerir poco mantenimiento y monitoreo ya que no se cuenta con un equipo disponible para estar vigilando el proceso constantemente.

- El sistema deberá generar alertas de rendimiento o fallos, estas alertas deberán ser fácilmente monitoreadas.

- El sistema deberá permitir desplegar y mantener varios modelos de diferentes tipos a la vez y no limitarse sólo a créditos.

- El sistema deberá acoplarse en la mayor medida a los conocimientos técnicos con lo que ya cuenta el equipo de analítica y requerir la menor cantidad posible de entrenamiento o aprendizaje de nuevas herramientas o lenguajes.

3.5 ANALISIS DE SOLUCIONES EXISTENTES

Con el fin de determinar cuál es la mejor solución se analizó las distintas ofertas disponibles, paralelamente se calificó cada una de ellas de acuerdo con los criterios definidos en la metodología con el fin de acercarse a los requisitos funcionales y no funcionales licitados.

A continuación, para cada una de las cinco ofertas disponibles se responderá:

- ¿Qué es?
- ¿Cómo funciona?
- Análisis de cada una de las variables

3.5.1 Clipper

- ¿Qué es?

Clipper es un servicio de baja latencia para machine learning que permite integrar distintos modelos analíticos con sistemas que son de cara al usuario.

- ¿Cómo funciona?

Un clúster de Clipper funciona mediante una colección de dockers que se comunican entre sí, los distintos comandos durante la configuración son administrados por Clipper admin tools, desde allí se puede crear o destruir dockers. Clipper es agnóstico de los modelos y sólo se concentra en su despliegue. Cada modelo es almacenado en un Docker, cuando ocurre el despliegue se levanta toda la colección de dockers asociada y se crea una API REST interna mantenida por el Management Front end.

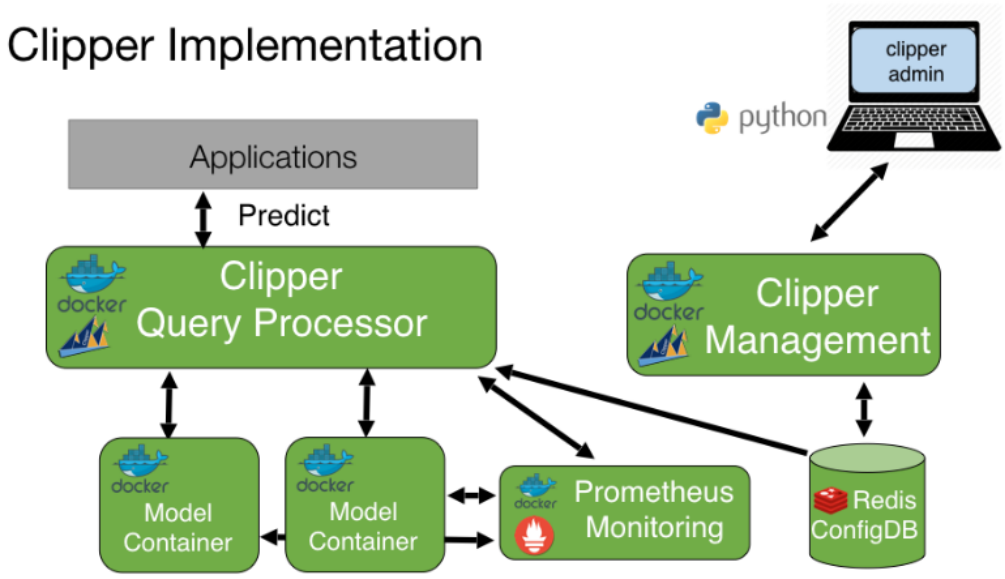


Figura 20: Implementación de Clipper. Extraido de: http://clipper.ai/tutorials/basic_concepts/

- Análisis de variables:

  - Disponibilidad: Clipper por su cuenta no ofrece una solución especifica al respecto, esto depende del servidor donde esté alojado cuya configuración y especificaciones se realizan aparte.

  - Escalabilidad: Clipper es fácilmente escalable ya que cuenta con múltiples opciones para establecer rangos de demanda y generar réplicas de los modelos, de esta forma Clipper actúa como un balanceador de carga para cada uno de ellos, permitiéndole satisfacer desde demandas muy pequeñas hasta demandas muy grandes

  - Facilidad de implementación: La implementación de Clipper requiere conocimientos medio-avanzado de Docker, nadie del equipo de analítica cuenta con este conocimiento por lo cual se considera tiene una implementación compleja

  - Despliegue: Este es un punto fuerte de Clipper ya que cuenta con un componente REST API fácilmente administrado el cual permite realizar peticiones tipo POST todo esto administrado desde el Clipper admin.

- Compatibilidad: En la arquitectura de la Fintech existen componentes que funcionan en Docker con buenos resultados, su integración con la arquitectura existente es totalmente posible, aunque no es sencilla.

- Mantenimiento y monitoreo: El mantenimiento y monitoreo se realiza a través de un servidor prometheus, este se inicia automáticamente al iniciar un clúster de Clipper y mediante el uso de herramientas como grafana se podría visualizar estas alertas. Si bien Clipper cuenta con unas métricas de rendimiento por defecto también es posible utilizar métricas definidas por el usuario.

- Concurrencia: La capacidad de manejar múltiples peticiones durante el día depende de la capacidad del servidor donde se encuentre alojado, se destaca la replicación de modelos.

- Modelos: Debido a que es agnóstico puede soportar una gran cantidad de tipos de modelos tanto en Python como en R.

Tabla 1

Puntuación obtenida en las distintas variables por Clipper.

Variable	Puntaje
Disponibilidad	2
Escalabilidad	3
Implementación	1
Despliegue	3
Compatibilidad	2
mant y monit	2
Concurrencia	2
Modelos	3
Total	18

3.5.2 Rorodata

- ¿Qué es?

  Rorodata se define como una PaaS (Platform as a service) diseñada para que los científicos de datos puedan construir, desplegar y monitorear modelos en producción.

- ¿Cómo funciona?

  Rorodata es una plataforma totalmente en la nube, su interacción se realiza mediante Jupyter Lab y provee servicios desde la construcción del modelo hasta el proceso de reentrenamiento, a continuación, se ve un diagrama con los distintos servicios

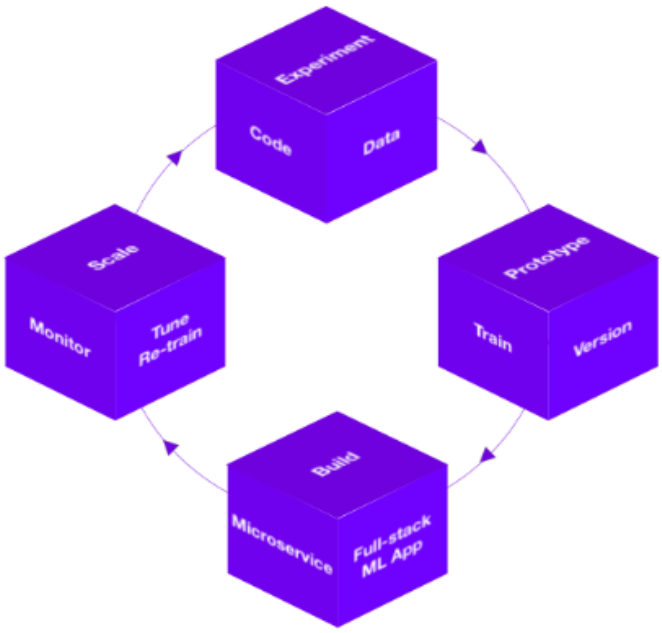


Figura 21: Servicios disponibles al utilizas rorodata. Extraído de : https://rorodata.com/

- Análisis de variables:

    - Disponibilidad: Al igual que Clipper, por su cuenta no ofrece una solución especifica al respecto. Sin embargo, Rorodata es mucho más flexible durante la instanciación de un servidor, ofrecen distintas posibles configuraciones de acuerdo con un rango de precios. Al estar en la nube deberían en teoría garantizar una disponibilidad constante, pero no se tiene más detalles sobre la infraestructura que están utilizando.

    - Escalabilidad: La escalabilidad de Rorodata depende del análisis del monitoreo y no es completamente automático.

    - Facilidad de implementación: Es de fácil implementación, no requiere conocimientos muy específicos en alguna herramienta o lenguaje, está construido para un uso pensado en Python.

    - Despliegue: Una vez instalado rorodata, el despliegue no es complicado, tan sólo requiere de unas especificaciones en un archivo yml y unos pocos comandos desde la terminal. Cuentan con una solución propia para la creación de una API a ser consumida.

    - Compatibilidad: Permite ciertas conexiones interesantes a través de plugin, sin embargo, debido al poco tiempo desde su lanzamiento todavía se espera el lanzamiento de varios más. Aun así, se podría integrar con la arquitectura ya existente

    - Mantenimiento y monitoreo: No ofrecen una solución especifica o propia para el monitoreo, rorodata se limita a la generación de un conjunto de logs sobre cada modelo, la implementación de algún sistema que permita el análisis de estos con facilidad depende de cada usuario.

    - Concurrencia: La cantidad de peticiones que puede soportar depende de la configuración que se realice desde Jupyter Lab la cual es muy flexible, no cuenta con un balanceador de carga propio y el usuario debería de encargarse de esa parte.

    - Modelos: Utiliza Jupyter para la construcción de los modelos por lo cual se cuenta con los diferentes tipos disponible para Python, es importante recordar que Jupyter también permite el uso del kernel de R pero debido a que no se encuentra en un servidor propio se desconoce si el kernel se reinicia cada que se reinicia el notebook como sucede comúnmente.

Tabla 2

Puntuación obtenida en las distintas variables por Rorodata.

Variable	Puntaje
Disponibilidad	3
Escalabilidad	2
Implementación	3
Despliegue	3
Compatibilidad	3
mant y monit	1
Concurrencia	2
Modelos	2
Total	19

3.5.3 Plumber

- ¿Qué es?

  Es un paquete que permite convertir cualquier código de R en una API web de manera sencilla

- ¿Cómo funciona?

  Una vez se tiene el código en R se procede a instalar el paquete de Plumber y mediante el uso de comentarios especiales se especifica que partes del código serán accesibles desde peticiones hacia el API REST y de qué manera.

- Análisis de variables:

  - Disponibilidad: La configuración y administración del servidor es responsabilidad única del usuario que implementa Plumber

  - Escalabilidad: Al igual que la disponibilidad no existe a través de Plumber nada que ofrezca una solución al respecto.

- Facilidad de implementación: Es muy fácil de implementar, posiblemente el más fácil de todas las opciones expuestas.

- Despliegue: Es el punto fuerte de Plumber, con muy pocas líneas de código se genera una API web capaz de comunicarse con el código creado en R.

- Compatibilidad: La compatibilidad con la arquitectura existe depende totalmente de una implementación completa del ambiente de R en la Fintech. Actualmente se dispone de hace muy poco de un servidor dedicado para esta plataforma el cual no ha sido probado debidamente.

- Mantenimiento y monitoreo: No ofrece ninguna solución al respecto, el equipo de analítica debería construir todo un esquema de captura de logs y visualización alrededor para manejar esto.

- Concurrencia: Depende enteramente de la configuración del servidor, R no es multihilo por lo tanto ante múltiples peticiones se debería construir un balanceador de carga que las encole y esperar que R vaya procesando una a una.

- Modelos: Cuenta con una amplia variedad de modelos disponibles que se encuentran para R a través de librerías como caret. Sin embargo, a diferencia de otras soluciones expuestas Plumber no está pensado para desplegar modelos en otros lenguajes por lo que supone una desventaja importante.

Tabla 3

Puntuación obtenida en las distintas variables por Plumber.

Variable	Puntaje
Disponibilidad	1
Escalabilidad	1
Implementación	3
Despliegue	3
Compatibilidad	2
mant y monit	1
Concurrencia	1
Modelos	2
Total	14

3.5.4 Flask

- ¿Qué es?

  Flask es un micro-framework web para Python mediante el cual se pueden crear sitios webs complejos o pequeñas APIS

- ¿Cómo funciona?

  Flask es considerado un microframework porque no requiere herramientas o librerías particulares adicionales. Dependiendo de la necesidad se acoplan distintos tipos de librerías necesarias a Flask para construir el proyecto.

- Análisis de variables:

  - Disponibilidad: Al igual que Plumber, la configuración y administración del servidor es responsabilidad única del usuario que lo implementa

  - Escalabilidad: Existen librerías complemento para Flask como REST-Plus o el despliegue en servidores de apache para lograr esto, sin embargo, la solución no es trivial.

  - Facilidad de implementación: Por si sólo el framework no es complejo, sin embargo, una solución enfocada al despliegue de machine learning en producción es un reto para el equipo de analítica a pesar de estar familiarizados con Python.

  - Despliegue: Flask permite el manejo de peticiones mediante la arquitectura REST, el despliegue de alguna API para el caso no es complicado y está ampliamente documentado.

  - Compatibilidad: El equipo de infraestructura de la Fintech ya cuenta con todo un esquema para desplegar este tipo de aplicaciones en Python por lo cual su integración no debería ser complicada.

  - Mantenimiento y monitoreo: Existe un conjunto de herramientas interesantes que permiten hacer esto. Hay librerías para Flask dedicadas únicamente al monitoreo además es fácilmente integrable con el kit de ELK.

- Concurrencia: Debido a la popularidad del framework existe una gran cantidad de información sobre cómo mejorar esta variable, sin embargo, al igual que plumber siempre va a depender de la configuración del servidor donde se encuentre alojado.

- Modelos: Flask está disponible en Python únicamente, a través de Scikit-Learn y demás librerías es posible acceder a un conjunto muy amplio de modelos, pero al igual que Plumber se encuentra limitado al lenguaje.

Tabla 4

Puntuación obtenida en las distintas variables por Flask.

Variable	Puntaje
Disponibilidad	1
Escalabilidad	2
Implementación	1
Despliegue	3
Compatibilidad	3
mant y monit	2
Concurrencia	2
Modelos	2
Total	16

3.5.5 SageMaker

- ¿Qué es?

  Sagemaker es una solución de Amazon, fue lanzada a finales del 2017 y está pensada para con el fin de permitirle a los científicos de datos, desarrollar, entrenar, optimizar, desplegar y monitorear una amplia variedad de modelos.

- ¿Cómo funciona?

  Sagemaker funciona principalmente a través de instancias de notebooks de Jupyter en los cuales vienen ya preparados ambientes para trabajar con TensorFlow, Pytorch, MXNet, R , etc. Cuenta además con algoritmos integrados de alto desempeño tales como xgboost, k-means o PCA los cuales (según Amazon) ofrecen un desempeño diez veces superior comparado con otras implementaciones.

- Análisis de variables:

    - Disponibilidad: Sagemaker es muy robusto, cuenta con el respaldo de Amazon y al estar en la nube no depende de un servidor físico aislado. Si el servidor se cae Amazon se encarga de reasignar los recursos que sean necesarios, siempre cuenta con múltiples respaldos.

    - Escalabilidad: Sagemaker cuenta con una característica conocida como auto-scaling. Esta consiste en aumentar o disminuir los recursos que sean necesarios automáticamente de acuerdo con la demanda del momento. Los limites en los que se puede escalar pueden ser definidos por el usuario con el fin de evitar sobrecostos.

    - Facilidad de implementación: Sagemaker se administra a través de notebooks de Jupyter principalmente, los algoritmos más populares se pueden acceder a través de un SDK que dispone Amazon, a menos de que se requiera un algoritmo muy específico que no esté dentro de las opciones de Sagemaker entonces se tendría que recurrir al uso de Dockers. No requiere conocimientos específicos amplios en algún lenguaje o herramienta que no conozca el equipo de analítica de la Fintech.

    - Despliegue: Este es uno de los aspectos más interesantes de Sagemaker ya que permite el despliegue de modelos con un solo clic. Al desplegarlo se crea un endpoint preparado para recibir peticiones REST tipo POST.

    - Compatibilidad: Debido a que es un servicio de Amazon, este ofrece una interoperabilidad completa con los demás servicios de la compañía, de esta forma se acopla perfectamente a la arquitectura ya existente en la Fintech.

    - Mantenimiento y monitoreo: Sagemaker se encuentra conectado con cloudwatch de Amazon, en este se puede programar un gran número de alertas y realizar un monitoreo avanzado. La visualización es fácil y la trazabilidad de error se realiza mediante logs que quedan almacenados en la historia de cloudwatch.

    - Concurrencia: Al igual que rorodata, sagemaker permite elegir distintas configuraciones de la maquina en la que se trabaja a través de múltiples rangos de precio. Sin embargo, debido a su característica de auto-scaling, la herramienta es capaz de soportar una cantidad de peticiones al día muy alta.

- Modelos: Los modelos más populares se encuentran optimizados y disponibles para su uso en Sagemaker, se puede trabajar tanto en Python como en R. Si el modelo no estuviese disponible se puede traer s sagemaker con toda la configuración de ambiente necesario utilizando Docker.

Tabla 5

Puntuación obtenida en las distintas variables por SageMaker.

Variable	Puntaje
Disponibilidad	3
Escalabilidad	3
Implementación	3
Despliegue	3
Compatibilidad	3
mant y monit	3
Concurrencia	3
Modelos	3
Total	24

3.5.6 Puntuación final de cada una de las opciones analizadas

Luego de realizar un análisis de las siete variables para cada una de las cinco posibles soluciones, se presenta a continuación el consolidado de los puntajes:

Servicio	Disponibilidad	Escalabilidad	Implementacion	Despliegue	Compatibilidad	mant y monit	Concurrencia	Modelos	Total
Clipper	2	3	1	3	2	2	2	3	18
Rorodata	3	2	3	3	3	1	2	2	19
Plumber	1	1	3	3	2	1	1	2	14
Flask	1	2	1	3	3	2	2	2	16
Sagemaker	3	3	3	3	3	3	3	3	24

Figura 22: Puntuación final de cada una de las opciones analizadas

Cómo se observa la herramienta que mejor puntaje tuvo fue Amazon Sagemaker, seguida de Rorodata cuya mayor debilidad fue por el mantenimiento y monitoreo. Por lo anterior se procederá a implementar el modelo creado en Sagemaker.

3.6 IMPLEMENTACIÓN DE SAGEMAKER

Para describir el proceso de implementación de Sagemaker en la Fintech se explicarán las distintas etapas donde interviene la herramienta en la construcción de un modelo analítico, estas fases son: desarrollar, entrenar y optimizar, desplegar y monitorear.

A continuación, se muestra un diagrama que ilustra como entrenar y desplegar un modelo:

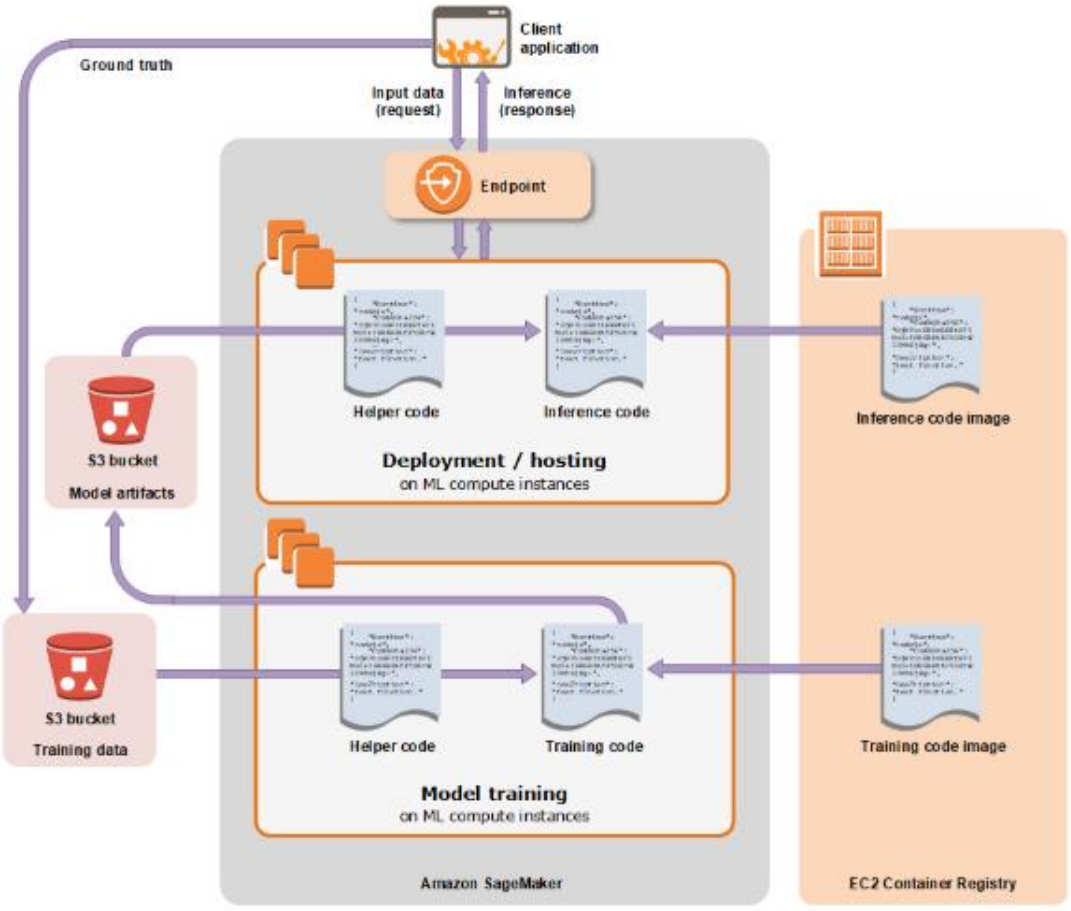


Figura 23: Diagrama de implementación de sagemaker. Extraído de: https://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Como se observa en el diagrama, junto con Sagemaker interviene otros dos servicios de Amazon: S3 que es donde se va a almacenar los datos para el entrenamiento y el modelo una vez se termine el entrenamiento, para este caso creo un bucket dedicado al modelo de analítica el cual contiene las particiones de datos train, test y validate creadas a partir de scikit-learn. EC2 se encarga de proveer poder computacional para el entrenamiento y mantener el despliegue. Posteriormente la aplicación se encarga de realizar peticiones a la espera de las inferencias que obtenga del endpoint.

3.6.1 Configuraciones Iniciales

Los procedimientos necesarios para la creación e inscripción en Amazon no se abordarán en este trabajo ya que la Fintech cuenta con varios servicios contratados con ellos por lo cual ya existe un procedimiento alrededor de esto y no difiere para Sagemaker.

3.6.2 Desarrollo

Una vez se cuente con el bucket de S3 y las credenciales en Amazon se procede a crear una instancia de un notebook desde la consola de Sagemaker

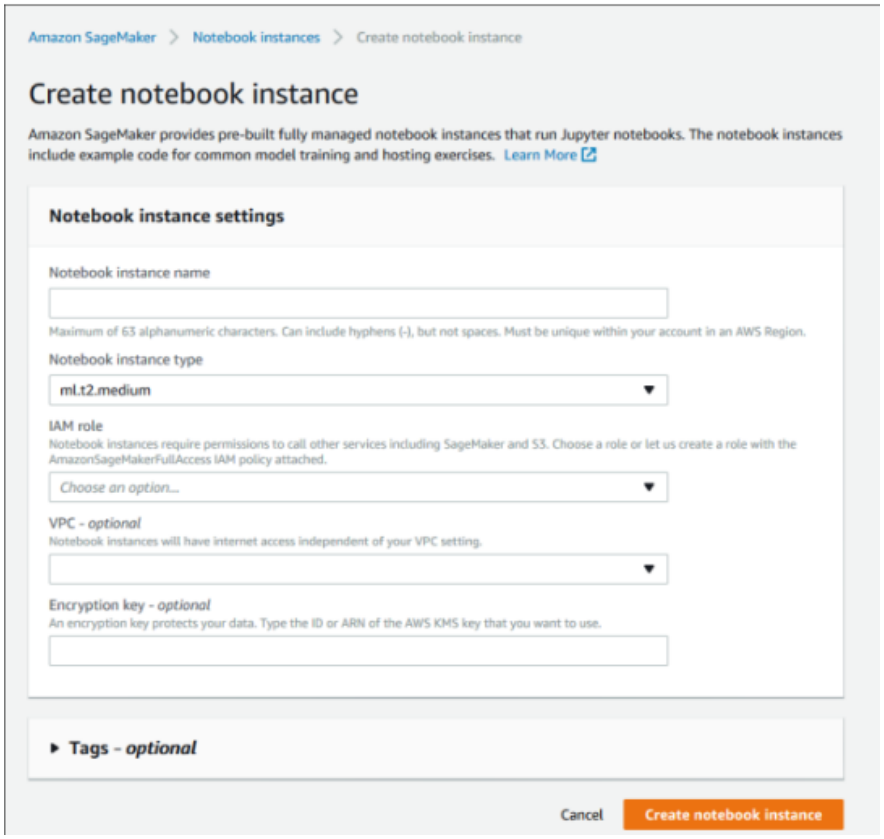


Figura 24: Creación de una instancia en SageMaker

Notebook instance type se refiere al tipo de EC2 que se utilizará, este notebook todavia no hace referencia al ambiente de trabajo para el desarrollo del modelo, solamente a su configuración. Para el caso de la fintech no se requiere ninguna VPC especial, el iam role es administrado y suministrado por el equipo de infraestructura y la ml.t2.medium es suficiente para nuestro modelo.

Una vez se crea la instasncia de EC2 podemos ingresar a la instancia del notebook de Jupyter. El modelo que se utilizó para la fintech es xgboost ya que se contaba con una carga desbalanceada,gran cantidad de nulos y variables de compleja normalización y el modelo maneja de forma adecuada esto.

Una vez dentro del notebook de Jupyter y luego de realizar las respectivas cargas de los datos y transformaciones , se llama al algoritmo:

```
from sagemaker.amazon.amazon_estimator import get_image_uri

sess = sagemaker.Session()

container = get_image_uri(region, 'xgboost')

xgb = sagemaker.estimator.Estimator(container,
                                    role,
                                    train_instance_count=1,
                                    train_instance_type='ml.m4.xlarge',
                                    output_path='s3://{}/{}/output'.format(bucket, prefix),
                                    sagemaker_session=sess)

xgb.set_hyperparameters(eval_metric='auc',
                        objective='binary:logistic',
                        num_round=100,
                        rate_drop=0.3,
                        tweedie_variance_power=1.4)
```

Figura 25: creación de Xgboost en SageMaker un para un código ampliado ver anexo 1

El mayor reto del uso de este modelo consiste en la optimización de sus hiperparametros los cuales tradicionalmente se encuentran a través de mallas de busqueda requiriendo un muy alto poder computacional. Todos los llamados a los distintos servicios de Amazon se realizan a través de Boto3.

3.6.3 Entrenamiento y optimización

Luego de haber instanciado el modelo se debe proceder con el entrenamiento y la optimización de los hiperparametros para ello Sagemaker cuenta con una característica conocida como HPO.

Para realizar la optimización de hiperparametros de nuestro modelo mediante esta característica se siguieron los siguientes pasos:

- Se establece el rango de hiperparametros con el que se trabajará en la optimización, estos serán avalados utilizando la métrica del área bajo la curva (AUC) que entregue el modelo con los datos dispuestos para la validación. Para este caso se utilizó los intervalos por defecto para el modelo:

```
hyperparameter_ranges = {'eta': ContinuousParameter(0, 1),
                         'min_child_weight': ContinuousParameter(1, 10),
                         'alpha': ContinuousParameter(0, 2),
                         'max_depth': IntegerParameter(1, 10)}
```

Figura 26: Configuración de hiperparametros

- Se especifica que la metrica utilizada para elegir la mejor combinación de hiperparametros será la AUC con el set de validación.

- Se crea un objeto de la clase HyperparameterTuner al cual se le especifica que se corran tres jobs en paralelo con la optimización y que como máximo corra veinte jobs, estos valores son un estandar de la herramienta

- Se lanza una maquina EC2 con esta configuración para que inicie el proceso de optimización, este puee durar alrededor de 30 minutos

- Utilizando Boto se describe el estado del servicio, si el proceso no ha terminado lo notifica, si ya terminó devuelve el AUC logrado y el set con la combinación más adecuada, además luego de hacer esto se guarda en s3 los model-artifacts del mejor modelo, listos para realizar el despliegue tal y como se ilustró anteriormente.

3.6.4 Despliegue y mantenimiento

El despliegue del modelo puede realizarse desde la consola de Sagemaker o utilizando el SDK para Python que dispone Amazon.

Para desplegar el modelo crediticio tan sólo se debe usar el método *deploy* perteneciente al modelo creado y especificar qué tipo de instancia EC2 que se utilizará. Esto generara un endpoint el cual puede ser administrado desde la consola, con este endpoint se puede proceder a enviar peticiones uno a uno o un gran número a la vez gracias a la característica Batch Transform.

Como se mencionó en las características de Sagemaker, este sistema se puede escalar infinitamente de forma automática, para escalarlo sólo hay que dirigirse a la consola y en la pestaña de endpoints seleccionamos el endpoint del modelo a escalar posteriormente la opción *configure auto scaling* para nuestro caso se seleccionó un número mínimo de instancias de tres y uno máximo de cinco.

Para probar el endpoint basta con realizar una petición HTTP tipo POST, debido a que Amazon tiene restricciones de acceso no es suficiente con realizar un Curl se deben especificar credenciales para poder realizar esto, la forma mediante la cual se realizó en la Fintech fue utilizando POSTMAN obteniendo buenos resultados.

El diagrama final de arquitectura para el consumo del modelo en producción quedó de la siguiente manera:

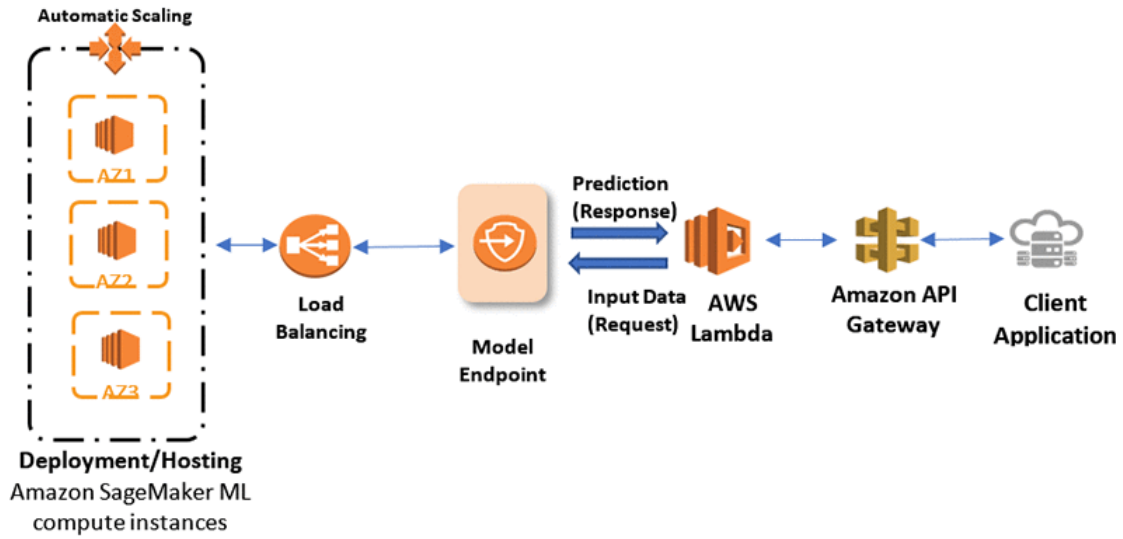


Figura 27: Diagrama final de la arquitectura escalable. Extraído de: https://aws.amazon.com/es/blogs/machine-learning/load-test-and-optimize-an-amazon-sagemaker-endpoint-using-automatic-scaling/

- Client Application: Se refiere a la aplicación móvil de la Fintech desde donde se realizarían las consultas de aprobación crediticia por parte de los usuarios.

- Amazon API Gateway: Es un orquestador que permite crear, publicar, mantener, monitorizar y proteger API a cualquier escala. Permite comunicar distintos servicios de Amazon que ser requieran desde una API tal y como se refleja en esta imagen:

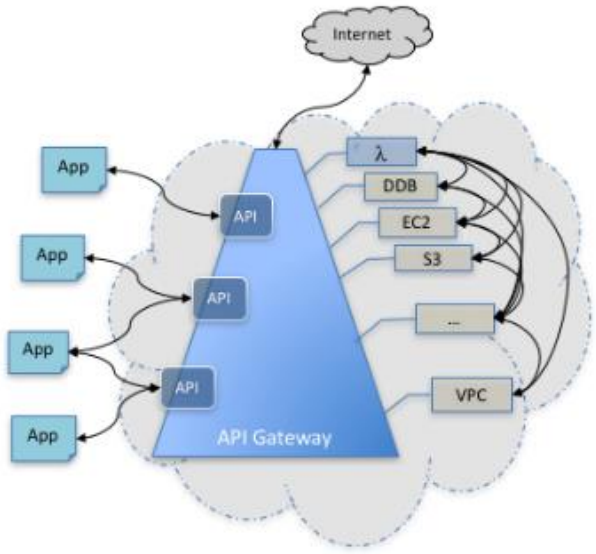


Figura 28: Funcionamiento de API Gateway. Extraído de: https://docs.aws.amazon.com/es_es/apigateway/latest/developerguide/welcome.html

En este escenario es el encargado de vincular las peticiones desde la aplicación del usuario con aws Lambda.

Es importante aclarar que la Fintech cuenta con un equipo dedicado exclusivamente a las APIs, este equipo tiene amplia experiencia en el uso de API Gateway, por lo cual el equipo de analítica no tiene necesidad de intervenir en este punto lo que no aumenta la complejidad en la implementación como se mencionó en el análisis de las distintas herramientas.

- AWS Lambda: Es un referente en la tecnología serverless, en este diagrama se encarga de recibir las peticiones de la API y enviarle esta información al endpoint en un formato específico, para nuestro caso es en text/csv. Una vez obtenga una respuesta del endpoint deberá remitirla a API Gateway para presentarla en la aplicación.

- Model endpoint: Es la dirección a la cual se apunta para realizar las peticiones POST, se administra desde la consola.

- Load balancing: es un balanceador de carga, se encarga de distribuir las peticiones entrantes entre las distintas instancias EC2 que se estén activas en ese momento. No requiere configuración adicional, Sagemaker lo hace automáticamente

- Deployment/Hosting: Son las instancias EC2 que se van escalando automáticamente de acuerdo con la demanda y lo establecido en la configuración

del endpoint. Es importante aclarar que Amazon garantiza la disponibilidad de estas y previene la caída de una de ellas cambiando la región activa en la que se encuentre.

3.6.5 Resultados del modelo crediticio

Luego de realizar el descriptivo estadístico se decidió utilizar sólo seis variables para el entrenamiento incluyendo la función objetivo-llamada “Grupo”, cero representa las malas pagas y uno las buenas pagas, a continuación, se puede ver un fragmento del set de entrenamiento:

:

	Grupo	Valor Gestion	Valor_saldo_cuenta	Cash in	Cash out	Transfer
0	0	0.0	2000.83	2	2	0
1	1	0.0	6904.17	24	20	7
2	0	0.0	1800.44	4	3	0
3	1	0.0	5014.89	0	2	0
4	0	0.0	1013.44	56	59	0

Figura 29: Set final a usar en el entrenamiento.

No se contaba con una muestra totalmente balanceada, a continuación, se muestra la distribución de clases:

Grupo	
0	0.57342
1	0.42658

Figura 30: Proporción de grupos en el set de datos

Cómo se mencionó anteriormente, para este modelo y en general para cualquier modelo analítico que se realice en la Fintech basado en el uso, siempre va a existir el mismo problema y es la gran cantidad de datos faltantes debido al poco uso de la aplicación por parte de los usuarios, a continuación, se presenta la distribución de densidad de las variables que se utilizaron:

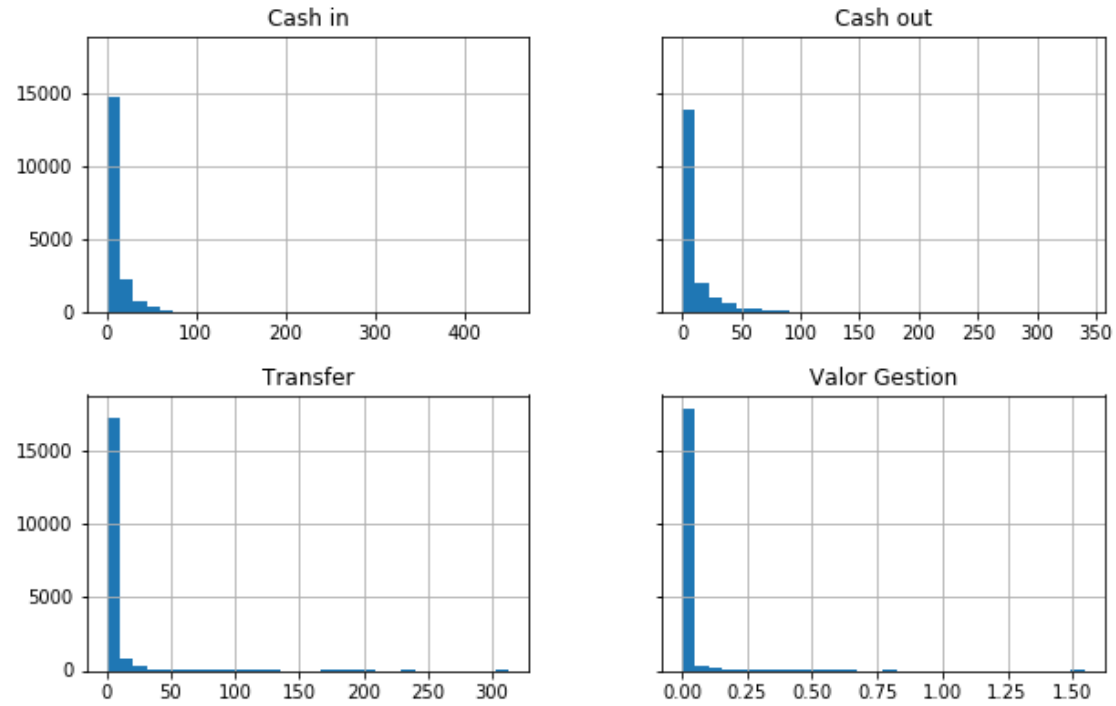


Figura 31: Distribución de las distintas variables en el set de datos

Como se puede observar todas ellas se encuentran sesgadas a la derecha confirmando la falta de actividad por parte de los usuarios. Eliminar nulos o imputar valores no es una opción de ser así no se podría construir ningún modelo basado en el uso.

Durante el proceso se encontró una correlación muy alta entre dos variables que para el negocio son totalmente distintas estas son cash-in y cash-out como se observa a continuación:

	Valor Gestion	Valor_saldo_cuenta	Cash in	Cash out	Transfer
Valor Gestion	1.000000	0.814033	0.113481	0.070128	0.063130
Valor_saldo_cuenta	0.814033	1.000000	0.127580	0.095353	0.078869
Cash in	0.113481	0.127580	1.000000	0.798681	0.389223
Cash out	0.070128	0.095353	0.798681	1.000000	0.308476
Transfer	0.063130	0.078869	0.389223	0.308476	1.000000

Figura 32: Correlacion entre las variables del set de datos

Una relación tan alta entre estas dos variables confirma el poco uso ya que confirma un tránsito muy rápido del dinero en la aplicación.

La calificación general del modelo resultó ser de 0.72 AUC. En cuanto a precisión se obtuvo la siguiente matriz de confusión:

predictions	0.0	1.0
actuals		
0	881	182
1	464	331

Figura 33: Matriz de confusión entregada por el modelo.

De lo anterior se obtiene que el modelo predijo bien 1212 casos de 1858 lo que se traduce en una precisión del 65%, 0.82 para los malos y tan sólo 0.41 para los buenos. Aunque es una precisión baja hay que tener en cuenta que se debe realizar una comparación el modelo de Lenddo, si bien este no agrupó los grupos de riesgo es posible comparar su calificación agrupando las bandas a +1 y -1 en el error, al hacer esto se puede obtiene una precisión del 63.46% menor a la obtenida por el modelo basado en uso creado.

3.7 VALIDACIÓN

El proceso de validación del modelo consta de dos fases: la primera es validar la implementación del modelo y la segunda es la validación de los resultados utilizando la escala descrita en la metodología sobre los requisitos no funcionales para la primera fase y los funcionales para la segunda.

3.7.1 Requisitos no funcionales

- El sistema deberá tener alta disponibilidad ya que una persona puede solicitar un crédito en cualquier momento.

  - Calificación: 4

  - Justificación: Debido a que el sistema implementado cuenta con el respaldo de Amazon, este provee servidores back-ups en distintas regiones lo que reduce la probabilidad de una falta de servicio, sin embargo, no se otorga un 5 por que no existe un sistema con disponibilidad perfecta.

- El sistema deberá ser fácilmente escalable en cuanto a su capacidad de procesamiento.

  - Calificación: 5

  - Justificación: Como se describió en la implementación, Sagemaker no sólo es escalable, sino que además lo hace automáticamente pudiendo incluso ser teóricamente de manera infinito bajo el supuesto de recursos ilimitados.

- El tiempo de respuesta no deberá ser mayor a 5 minutos.

  - Calificación: -

  - Justificación: El tiempo de respuesta está más condicionado por el tipo de modelo que por la arquitectura, aun así, un tiempo de 5 minutos es demasiado alto según lo observado en pruebas durante la implementación del endpoint donde las respuestas estaban en el orden de milésimas o centésimas. No se otorga ninguna calificación

- El sistema deberá ser capaz de manejar múltiples peticiones por día.

  - Calificación: 5

  - Justificación: No existe un límite de peticiones diarias al sistema, esto va a depender principalmente de la escalabilidad y la disponibilidad.

- El sistema deberá ser fácilmente adaptable a la arquitectura ya existente de la Fintech y el componente de API Gateway

  - Calificación: 5

  - Justificación: Debido a que la arquitectura de la Fintech se encontraba en su mayoría en Amazon, el sistema se acopla perfectamente y al ser de la misma empresa ofrece facilidades que no serían factibles mediante otro sistema.

- El sistema debe requerir poco mantenimiento y monitoreo ya que no se cuenta con un equipo disponible para estar vigilando el proceso constantemente.

  - Calificación: 5

  - Justificación: Sagemaker es fácilmente integrable con Amazon Cloudwatch facilitando su monitoreo. En cuanto al mantenimiento, no se requiere ningún mantenimiento en cuanto al servidor ya que esto lo realiza Amazon.

- El sistema deberá generar alertas de rendimiento o fallos, estas alertas deberán ser fácilmente monitoreadas.

- Calificación: 5

- Justificación: Como se mencionó en el requisito anterior, el sistema es fácilmente integrable con Cloudwatch por lo cual las alertas generadas se agregarían fácilmente al monitor de las demás alertas que vigila el equipo de infraestructura, Sagemaker reporta automáticamente los logs de todos los incidentes.

- El sistema deberá permitir desplegar y mantener varios modelos de diferentes tipos a la vez y no limitarse sólo a créditos.

- Calificación: 5

- Justificación: El sistema cuenta con varios algoritmos cuya implementación es nativa además permite el uso de cualquier otro algoritmo mediante el uso de Docker por lo cual no existe ninguna limitación en este sentido, incluso durante el desarrollo del proyecto se realizó una implementación de un modelo de fraude en recargas que corría paralelo al de créditos.

- El sistema deberá acoplarse en la mayor medida a los conocimientos técnicos con lo que ya cuenta el equipo de analítica y requerir la menor cantidad posible de entrenamiento o aprendizaje de nuevas herramientas o lenguajes.

- Calificación: 4

- Justificación: Sagemaker ofrece un acoplamiento con la arquitectura existente mediante el uso de tecnologías y lenguajes que ya conocía el equipo, sin embargo, para la creación de modelos muy específicos, es necesario el conocimiento de Docker el cual nadie en el equipo dominaba.

3.7.2 Requisitos funcionales

- Desde la aplicación el sistema deberá permitir a cada usuario conocer si su crédito fue aprobado o no.

• Calificación: 4

• Justificación: Mediante el uso del endpoint y API Gateway la integración con la aplicación es sencilla y se puede obtener una respuesta inmediata de cualquier petición

desde la aplicación. El modelo entrega esta aprobación con una precisión cercana al 65%, debido a que puede ser mejorado en trabajos futuros no se otorga la calificación máxima.

- El sistema entregará información acerca de la calificación otorgada a una persona a personas de negocio con el fin de evaluar casos específicos.

  - Calificación: 4

  - Justificación: El negocio debe establecer un umbral de aprobación del cual no existe ningún estudio hasta ahora por parte de ellos, pero una vez se cuente con él, el modelo está en toda la capacidad de redirigir esos casos específicos al sistema de BI de la Fintech conocido como Metabase, desde allí seria evaluado por el área de servicio.

- Si una persona que no ha solicitado un crédito es candidato a ser buena paga, el sistema deberá estar en la capacidad de identificarlo y entregar este insumo para generar recomendaciones en la aplicación.

• Calificación: 5

• Justificación: Como se mencionó anteriormente, el modelo pude procesar peticiones uno a uno o en batch, de esta forma se evaluarían distintos grupos de interés para el negocio.

De acuerdo con lo anterior de 55 puntos posibles (excluyendo el requisito de los cinco minutos) se obtuvo una calificación de 51 puntos lo cual refleja un cumplimiento de 92% de los requisitos.

4. CONCLUSIONES Y CONSIDERACIONES FINALES

Las conclusiones y consideraciones finales para la Fintech serán desarrolladas en dos partes, la primera corresponde a lo respectivo al modelo y la segunda al sistema correspondiente a su implementación.

4.1 ACERCA DEL MODELO

- Se desarrolló un modelo utilizando variables alternativas, para este caso el uso de los usuarios en la aplicación en cuanto su uso y gestión del dinero principalmente. Este modelo arrojó en su primera corrida arrojó una precisión del 65% frente al 63.46% de Lenddo teniendo en cuenta una muestra más representativa de los usuarios.

- En la Fintech, la construcción de modelos analíticos que se basen en el uso de la aplicación tiene el mismo problema y es la gran proporción de datos ausentes lo que lleva a resultados muy inconclusos.

- El modelo sugiere una tendencia de la Fintech como plataforma de tránsito. Esto quiere decir que el dinero entra es muy similar al que sale de la aplicación como demuestra la correlación de 0.79 entre las variables de cash-in y cash-out.

- Los modelos de crédito que utilizan variables alternativas son una excelente opción para aumentar la inclusión financiera y como complemento de los tradicionales más no como modelos principales.

- Lenddo es una herramienta con mayor potencial en entender el comportamiento de las personas, mediante segmentaciones basadas en los datos alternativos, que permitan acciones dirigidas que como proveedor de un modelo de otorgamiento.

- Debido a que la Fintech espera otorgar créditos conociendo la menor cantidad de información posible sobre los usuarios, se debe considerar la posibilidad de implementar un modelo crediticio basado en el incremento gradual de la confianza.

- Los datos alternativos para modelos de crédito deberían de ser mucho más robustos y contextuales tales como: cumplimiento de pago en facturas, historiales de pagos, ingresos, etc. Además, el uso de variables psicométricas es controvertido y manipulable.

4.2 ACERCA DEL SISTEMA

- La arquitectura de la Fintech está ampliamente implementada en Amazon, como lo demuestra la gran cantidad de servicios en la esta nube tales como: aws lambda, api Gateway, Redshift, Dynamo DB, etc.

- Luego de la revisión de las necesidades de la Fintech y las distintas opciones disponibles para el consumo e implementación del modelo la solución de Amazon, sagemaker, obtuvo la máxima calificación en todos los numerales evaluados.

- Durante la evaluación del cumplimiento de requisitos funcionales y no funcionales, Sagemaker obtuvo una calificación de cumplimiento del 92% cediendo puntos en disponibilidad (ya que no existe un sistema con disponibilidad perfecta), facilidad para la implementación desde los conocimientos del equipo (no se había usado antes).

- Con la propuesta presentada, la Fintech está en capacidad de adaptarse fácilmente a cualquier nivel de demanda sin incurrir en sobrecostos gracias al ajuste automático de Sagemaker.

- Se siguió un marco de trabajo que satisficiera las necesidades de la Fintech teniendo en cuenta los 4 elementos postulados por Chitipothu: administración, monitoreo, construcción y despliegue.

- Luego de la implementación de este proyecto, no sólo se implantó la arquitectura necesaria para el consumo del modelo crediticio generado sino para cualquier modelo analítico que se genere en la Fintech, un ejemplo de esto es la implementación durante el mismo periodo de trabajo de un modelo para detectar fraude en fraccionamiento de recargas en la plataforma.

- La arquitectura fue capaz de proveer la capacidad necesaria a la analítica sin sacrificar los demás procesos tecnológicos tal y como lo sugirió Kalechofsky (2016)

- Existe una gran cantidad de datos faltantes para muchos usuarios especialmente aquellos que se registraron en los primeros días de funcionamiento de la aplicación ya que no existían controles ni registros.

- En un futuro, la Fintech podría plantearse robustecer su arquitectura de datos de tal forma que procesos donde las fuentes de información no se obtengan directamente de la aplicación sino de un tercero puedan incluirse en el flujo para analítica en tiempo real y no en batch como sucede actualmente.

5. REFERENCIAS

Altman, E. I. (1968). FINANCIAL RATIOS, DISCRIMINANT ANALYSIS AND THE PREDICTION OF CORPORATE BANKRUPTCY. *The Journal of Finance*, *23*(4), 589–609. https://doi.org/10.1111/j.1540-6261.1968.tb00843.x

Bolton, C. (2010). Logistic regression and its application in credit scoring. Retrieved from https://repository.up.ac.za/handle/2263/27333

Brownlee, J. (2016). A Gentle Introduction to XGBoost for Applied Machine Learning. Retrieved October 15, 2018, from https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

Chitipothu Rorodata, A. (n.d.). Managing Machine Learning Models in Production. Retrieved from https://cdn.oreillystatic.com/en/assets/1/event/266/Managing machine learning models in production Presentation.pdf

Crankshaw, D., Wang, X., Zhou, G., Franklin, M. J., Gonzalez, J. E., Stoica, I., & Zhou, G. (n.d.). Clipper: A Low-Latency Online Prediction Serving System Clipper: A Low-Latency Online Prediction Serving System. Retrieved from https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/crankshaw

Crook, J. N., Edelman, D. B., & Thomas, L. C. (2007). Recent developments in consumer credit risk assessment. *European Journal of Operational Research*, *183*(3), 1447–1465. https://doi.org/10.1016/J.EJOR.2006.09.100

Decreto 2654 de 2014 Nivel Nacional. (n.d.). Retrieved May 21, 2018, from http://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=60260

Huang, C.-L., Chen, M.-C., & Wang, C.-J. (2007). Credit scoring with a data mining approach based on support vector machines. *Expert Systems with Applications*, *33*(4), 847–856. https://doi.org/10.1016/J.ESWA.2006.07.007

Kalechofsky, H. (2016). A Simple Framework for Building Predictive Models A Little Data Science Business Guide. Retrieved from http://www.msquared.com/wp-content/uploads/2017/01/A-Simple-Framework-for-Building-Predictive-Models.pdf

Lee, T.-S., Chiu, C.-C., Chou, Y.-C., & Lu, C.-J. (2006). Mining the customer credit using classification and regression tree and multivariate adaptive regression splines. *Computational Statistics & Data Analysis*, *50*(4), 1113–1130. https://doi.org/10.1016/J.CSDA.2004.11.006

Lee, Y., Scolari, A., Interlandi Microsoft Redmond, M., Weimer, M., & Chun, B.-G. (n.d.). Towards High-Performance Prediction Serving Systems. Retrieved from http://learningsys.org/nips17/assets/papers/paper_9.pdf

Lenoir, F. (n.d.). LENDDO: DRIVER OF FINANCIAL INCLUSION USING DIGITAL DATA. Retrieved from https://www.bceao.int/sites/default/files/inline-files/session_4_florentin_lenoir_lenddo_heure_14h00.pdf

Lessmann, S., Baesens, B., Seow, H.-V., & Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, *247*(1), 124–136. https://doi.org/10.1016/J.EJOR.2015.05.030

Min, J. H., & Lee, Y.-C. (2005). Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters. *Expert Systems with Applications*, *28*(4), 603–614. https://doi.org/10.1016/J.ESWA.2004.12.008

Mitchell, T. M. (n.d.). Machine Learning. Retrieved from https://www.cs.ubbcluj.ro/~gabis/ml/ml-books/McGrawHill - Machine Learning -Tom Mitchell.pdf

Mobile Phone Data as the Key to Promoting Financial Inclusion. (n.d.). Retrieved from https://www.wsbi-esbg.org/SiteCollectionDocuments/Cignifi.pdf

Nie, G., Rowe, W., Zhang, L., Tian, Y., & Shi, Y. (2011). Credit card churn forecasting by logistic regression and decision tree. *Expert Systems with Applications*, *38*(12), 15273–15285. https://doi.org/10.1016/J.ESWA.2011.06.028

Singh, R., & Rani, R. (2011). Comparative Evaluation of Predictive Modeling Techniques on Credit Card Data. *International Journal of Computer Theory and Engineering*, *3*(5). Retrieved from https://pdfs.semanticscholar.org/17f9/1346a5948b86ec2bbfc78675e9cfb83b311c.pdf

Universidad Nacional de Colombia. Departamento de Economía., O., Universidad Nacional de Colombia. Departamento de Teoría y Política Económica., O., & Universidad Nacional de Colombia. Escuela de Economía., C. V. (2013). *Cuadernos de economía : revista del Departamento de Economía, Universidad Nacional de Colombia. Cuadernos de Economía* (Vol. 32). Departamento de Economía, Universidad Nacional de Colombia. Retrieved from https://revistas.unal.edu.co/index.php/ceconomia/article/view/38348

West, D. (2000). Neural network credit scoring models. *Computers & Operations Research*, *27*(11–12), 1131–1152. https://doi.org/10.1016/S0305-0548(99)00149-5

Wiginton, J. C. (1980). A Note on the Comparison of Logit and Discriminant Models of Consumer Credit Behavior. *The Journal of Financial and Quantitative Analysis*, *15*(3), 757. https://doi.org/10.2307/2330408

6. ANEXO 1 – EJEMPLO DE IMPLEMENTACIÓN DE XGBOOST EN SAGEMAKER

```
bucket = '<your_s3_bucket_name_here>'
prefix = 'sagemaker/DEMO-xgboost-dm'

# Define IAM role
import boto3
import re
from sagemaker import get_execution_role

role = get_execution_role()
import numpy as np                                # For matrix
operations and numerical processing
import pandas as pd                               # For munging
tabular data
import matplotlib.pyplot as plt                   # For charts and
visualizations
from IPython.display import Image                 # For displaying
images in the notebook
from IPython.display import display               # For displaying
outputs in the notebook
from time import gmtime, strftime                 # For labeling
SageMaker models, endpoints, etc.
import sys                                        # For writing
outputs to notebook
import math                                       # For ceiling
function
import json                                       # For parsing
hosting outputs
import os                                         # For manipulating
filepath names
import sagemaker                                  # Amazon SageMaker's
Python SDK provides many helper functions
from sagemaker.predictor import csv_serializer    # Converts strings
for HTTP POST requests on inference
!wget https://archive.ics.uci.edu/ml/machine-learning-
databases/00222/bank-additional.zip
!unzip -o bank-additional.zip
data = pd.read_csv('./bank-additional/bank-additional-full.csv',
sep=';')
pd.set_option('display.max_columns', 500)     # Make sure we can see
all of the columns
```

```
pd.set_option('display.max_rows', 20)          # Keep the output on one
page
data
# Frequency tables for each categorical feature
for column in data.select_dtypes(include=['object']).columns:
    display(pd.crosstab(index=data[column], columns='% observations',
normalize='columns'))

# Histograms for each numeric features
display(data.describe())
%matplotlib inline
hist = data.hist(bins=30, sharey=True, figsize=(10, 10))
for column in data.select_dtypes(include=['object']).columns:
    if column != 'y':
        display(pd.crosstab(index=data[column], columns=data['y'],
normalize='columns'))

for column in data.select_dtypes(exclude=['object']).columns:
    print(column)
    hist = data[[column, 'y']].hist(by='y', bins=30)
    plt.show()
display(data.corr())
pd.plotting.scatter_matrix(data, figsize=(12, 12))
plt.show()
data['no_previous_contact'] = np.where(data['pdays'] == 999, 1, 0)
# Indicator variable to capture when pdays takes a value of 999
data['not_working'] = np.where(np.in1d(data['job'], ['student',
'retired', 'unemployed']), 1, 0)   # Indicator for individuals not
actively employed
model_data = pd.get_dummies(data)
# Convert categorical variables to sets of indicators
model_data = model_data.drop(['duration', 'emp.var.rate',
'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employed'],
axis=1)
train_data, validation_data, test_data =
np.split(model_data.sample(frac=1, random_state=1729), [int(0.7 *
len(model_data)), int(0.9 * len(model_data))])   # Randomly sort the
data then split out first 70%, second 20%, and last 10%
pd.concat([train_data['y_yes'], train_data.drop(['y_no', 'y_yes'],
axis=1)], axis=1).to_csv('train.csv', index=False, header=False)
pd.concat([validation_data['y_yes'], validation_data.drop(['y_no',
'y_yes'], axis=1)], axis=1).to_csv('validation.csv', index=False,
header=False)
boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(pref
ix, 'train/train.csv')).upload_file('train.csv')
boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(pref
ix, 'validation/validation.csv')).upload_file('validation.csv')
from sagemaker.amazon.amazon_estimator import get_image_uri
container = get_image_uri(boto3.Session().region_name, 'xgboost')
```

```
s3_input_train =
sagemaker.s3_input(s3_data='s3://{}/{}/train'.format(bucket, prefix),
content_type='csv')
s3_input_validation =
sagemaker.s3_input(s3_data='s3://{}/{}/validation/'.format(bucket,
prefix), content_type='csv')
sess = sagemaker.Session()

xgb = sagemaker.estimator.Estimator(container,
                                    role,
                                    train_instance_count=1,

train_instance_type='ml.m4.xlarge',

output_path='s3://{}/{}/output'.format(bucket, prefix),
                                    sagemaker_session=sess)
xgb.set_hyperparameters(max_depth=5,
                        eta=0.2,
                        gamma=4,
                        min_child_weight=6,
                        subsample=0.8,
                        silent=0,
                        objective='binary:logistic',
                        num_round=100)

xgb.fit({'train': s3_input_train, 'validation': s3_input_validation})
xgb_predictor = xgb.deploy(initial_instance_count=1,
                           instance_type='ml.m4.xlarge')
xgb_predictor.content_type = 'text/csv'
xgb_predictor.serializer = csv_serializer
def predict(data, rows=500):
    split_array = np.array_split(data, int(data.shape[0] / float(rows)
+ 1))
    predictions = ''
    for array in split_array:
        predictions = ','.join([predictions,
xgb_predictor.predict(array).decode('utf-8')])

    return np.fromstring(predictions[1:], sep=',')

predictions = predict(test_data.drop(['y_no', 'y_yes'],
axis=1).as_matrix())
pd.crosstab(index=test_data['y_yes'], columns=np.round(predictions),
rownames=['actuals'], colnames=['predictions'])
```

7. ANEXO 2 – EJEMPLO DE IMPLEMENTACIÓN DE HPO PARA XGBOOST EN SAGEMAKER

```
import sagemaker
import boto3
from sagemaker.tuner import IntegerParameter, CategoricalParameter,
ContinuousParameter, HyperparameterTuner

import numpy as np                                # For matrix
operations and numerical processing
import pandas as pd                               # For munging
tabular data
import os

region = boto3.Session().region_name
smclient = boto3.Session().client('sagemaker')

role = sagemaker.get_execution_role()

bucket = sagemaker.Session().default_bucket()
prefix = 'sagemaker/DEMO-hpo-xgboost-dm'
```

```
!wget -N https://archive.ics.uci.edu/ml/machine-learning-
databases/00222/bank-additional.zip
!unzip -o bank-additional.zip
```

```
DATA = PD.READ_CSV('./BANK-ADDITIONAL/BANK-ADDITIONAL-FULL.CSV', SEP=';')
PD.SET_OPTION('DISPLAY.MAX_COLUMNS', 500) # MAKE SURE WE CAN SEE ALL OF THE COLUMNS
PD.SET_OPTION('DISPLAY.MAX_ROWS', 50) # KEEP THE OUTPUT ON ONE PAGE DATA
```

```
data['no_previous_contact'] = np.where(data['pdays'] == 999, 1, 0)
# Indicator variable to capture when pdays takes a value of 999
data['not_working'] = np.where(np.in1d(data['job'], ['student',
'retired', 'unemployed']), 1, 0)   # Indicator for individuals not
actively employed
model_data = pd.get_dummies(data)
# Convert categorical variables to sets of indicators
model_data
```

```
model_data = model_data.drop(['duration', 'emp.var.rate',
'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employed'],
axis=1)
train_data, validation_data, test_data =
np.split(model_data.sample(frac=1, random_state=1729), [int(0.7 *
len(model_data)), int(0.9*len(model_data))])

pd.concat([train_data['y_yes'], train_data.drop(['y_no', 'y_yes'],
axis=1)], axis=1).to_csv('train.csv', index=False, header=False)
pd.concat([validation_data['y_yes'], validation_data.drop(['y_no',
'y_yes'], axis=1)], axis=1).to_csv('validation.csv', index=False,
header=False)
pd.concat([test_data['y_yes'], test_data.drop(['y_no', 'y_yes'],
axis=1)], axis=1).to_csv('test.csv', index=False, header=False)
boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(pref
ix, 'train/train.csv')).upload_file('train.csv')
boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(pref
ix, 'validation/validation.csv')).upload_file('validation.csv')
from sagemaker.amazon.amazon_estimator import get_image_uri

sess = sagemaker.Session()

container = get_image_uri(region, 'xgboost')

xgb = sagemaker.estimator.Estimator(container,
                                    role,
                                    train_instance_count=1,

train_instance_type='ml.m4.xlarge',

output_path='s3://{}/{}/output'.format(bucket, prefix),
                                    sagemaker_session=sess)

xgb.set_hyperparameters(eval_metric='auc',
                        objective='binary:logistic',
                        num_round=100,
                        rate_drop=0.3,
                        tweedie_variance_power=1.4)
hyperparameter_ranges = {'eta': ContinuousParameter(0, 1),
                        'min_child_weight': ContinuousParameter(1,
10),
                        'alpha': ContinuousParameter(0, 2),
                        'max_depth': IntegerParameter(1, 10)}
objective_metric_name = 'validation:auc'
```

```
tuner = HyperparameterTuner(xgb, objective_metric_name,
hyperparameter_ranges, max_jobs=20, max_parallel_jobs=3)
```

```
s3_input_train =
sagemaker.s3_input(s3_data='s3://{}/{}/train'.format(bucket, prefix),
content_type='csv')
s3_input_validation =
sagemaker.s3_input(s3_data='s3://{}/{}/validation/'.format(bucket,
prefix), content_type='csv')

tuner.fit({'train': s3_input_train, 'validation':
s3_input_validation})
boto3.client('sagemaker').describe_hyper_parameter_tuning_job(

HyperParameterTuningJobName=tuner.latest_tuning_job.job_name)['HyperPa
rameterTuningJobStatus']
```